

Novel approaches for automated data-processing in structure-guided drug design

Candidate: Yorgo EL MOUBAYED

Supervisors: Jose A. MARQUEZ and Raphael BOURGEAS

European Molecular Biology Laboratory (EMBL)
71 Avenue des Martyrs, 38000 Grenoble, France

Master: Bioinformatics

Major: Software development and data analysis

Year: 2020-2021

Acknowledgments

EMBL (<https://embl.fr/research/unit/marquez/index.html>)

Jose A. MARQUEZ (Head of the crystallization facility).

Raphael BOURGEAS (Postdoctoral fellow).

Peter MURPHY (Full stack developer).

Rahila RAHIMOVA (Postdoctoral fellow).

Nuria CIRAUQUI DIAZ (Postdoctoral fellow).

Cengizhan BUYUKDAG (Trainee).

Serena ROCCHIO (Postdoctoral fellow).

Veronica ZAMPIERI (Postdoctoral fellow).

Adeline ROBIN (Postdoctoral fellow).

Florine DUPEUX (Research engineer).

Anne-Sophie HUMM (Research technician).

Lea MAMMRI (Research technician).

EMBL-EBI (<https://www.ebi.ac.uk/about/people/sameer-velankar>)

John BERRISFORD (Scientific curator, PDBe-KB).

ALPX (<https://alpx-services.com/>)

Irina CORNACIU (CEO).

Damien CLAVEL (Research scientist).

Andrea PICA (Research scientist).

Global Phasing Limited (<https://www.globalphasing.com/>)

Gerard BRICOGNE (Director).

Clemens VONRHEIN (Software developer).

Andrew SHARFF (Software developer).

Aix-Marseille University

Aitor GONZALEZ (Headteacher).

James STURGIS (Professor/mentor).

Family and friends

Elias EL MOUBAYED.

Laurette EL MOUBAYED.

Perla EL MOUBAYED EJBEH.

Mario EL MOUBAYED.

Sara EL HAJJ (Predoctoral fellow, Aix-Marseille University).

Imad EJBEH.

List of abbreviations

- **ACID:** Atomicity, Consistency, Isolation, Durability.
- **ADMET:** Absorption, Distribution, Mechanism, Excretion, and Toxicity.
- **API:** Application Programming Interface.
- **CEO:** Chief Executive Officer.
- **CRIMS:** CRystallographic Information Management System.
- **CRUD:** Create, Read, Update, Delete.
- **CSV:** Comma-Separated Values.
- **DBMS:** Database Management System.
- **DNA:** DeoxyriboNucleic Acid.
- **DRF:** Django REST Framework.
- **EBI:** European Bioinformatics Institute.
- **EMBL:** European Molecular Biology Laboratory.
- **ESRF:** European Synchrotron Radiation Facility.
- **FAIR:** Findability, Accessibility, Interoperability, and Reusability.
- **FBDD:** Fragment-Based Drug Discovery.
- **GDP:** Guanosine DiPhosphate.
- **GPhL:** Global Phasing Limited.
- **GTP:** Guanosine TriPhosphate.
- **HTS:** High Throughput Screening.
- **HTTP:** HyperText Transfer Protocol.
- **HTX:** High Throughput crystallization.
- **IC₅₀:** half maximal inhibitory concentration.
- **ISPyB:** Information System for Protein crystallography Beamlines.
- **JSON:** JavaScript Object annotation.
- **M_w:** Molecular Weight.
- **mM:** milliMolar.
- **nM:** nanoMolar.
- **NS3:** Non-Structural protein 3.
- **OGM:** Object Graph Mapping.
- **ORM:** Object-Relational Mapping.

- **PDBe-KB:** Protein Data Bank in Europe Knowledge Base.
- **PDB:** Protein Data Bank.
- **QSAR:** Quantitative Structure-Activity Relationship.
- **REST:** REpresentational State Transfer.
- **RNA:** RiboNucleic Acid.
- **SAR:** Structure-Activity Relationship.
- **SLS:** Swiss Light Source.
- **SOA:** Service-Oriented Architecture.
- **SQL:** Structured Query Language.
- **TM:** Trademark.
- **URL:** Uniform Resource Locator.
- **UX:** User eXperience.

List of figures

- **Figure 1:** Schematic of the structure-based drug discovery pipeline.
- **Figure 2:** Fragment-based drug discovery.
- **Figure 3:** Workflow for solving molecular structures by X-ray crystallography.
- **Figure 4:** Fragments binding to the RAS protein.
- **Figure 5:** Discovery of a novel binding site in Hepatitis C virus NS3.
- **Figure 6:** Schematic of Pipedream's workflow.
- **Figure 7:** Schematic of the concept of online crystallography.
- **Figure 8:** CRIMS service-oriented architecture (SOA) structural components.
- **Figure 9:** Schematic of the project's development strategy.
- **Figure 10:** Trade-offs of different software architecture patterns.
- **Figure 11:** Underlying microservices of each CRIMS service.
- **Figure 12:** CRIMS SOA new structural components.
- **Figure 13:** Agile user stories collected from stakeholders.
- **Figure 14:** Modelling data as a property on a node.
- **Figure 15:** Modelling data as separate nodes.
- **Figure 16:** Graph data model of the novel data processing service.
- **Figure 17:** Database layer specifications of the novel data processing service.
- **Figure 18:** Layers and interactions of the novel data processing service.
- **Figure 19:** Data-driven testing for handling the registration of data processing settings.
- **Figure 20:** Schematic of Pipedream's decision tree.

Table of contents

<i>Acknowledgments</i>	3
<i>List of abbreviations</i>	5
<i>List of figures</i>	9
1. Introduction	13
1.1. Drug discovery	13
1.2. Identification of chemical leads	15
1.3. X-ray based high-throughput fragment screening	17
1.4. Structural data processing, analysis, and integration	19
1.5. High-throughput screening at EMBL Grenoble	25
1.6. Project outline	27
1.7. Project development strategy	31
2. Materials and methods	33
2.1. Architecture pattern	33
2.2. Pattern topology	33
2.3. Framework	35
3. Results	39
3.1. CRIMS novel SOA	39
3.2. Graph data model construction	39
3.3. Data access layer implementation	47
4. Discussions	49
5. Conclusions and perspectives	53
6. References	57
7. Supplementary data	67
<i>Abstract</i>	71

1. Introduction

1.1. Drug discovery

Drug discovery is the process of identifying probable new therapeutic entities. It combines the use of computational, experimental, translational tools, and clinical models. Despite advances in understanding biological systems at a molecular level, drug discovery remains a lengthy, costly and complex process with high attrition rates. [1].

An essential step in drug discovery is finding new chemical leads. A chemical lead is a molecule with good potency in biological assays that reflect the targeted mechanism. The lead is a molecule used in cell assays and model systems to validate a particular target's druggability and therapeutic potential. Still, it may not have all the characteristics of a drug. A drug must achieve high and sustained plasma concentrations after oral dosing to exert the desired effect. The atomicity, consistency, isolation, and durability (ADMET) of a chemical lead will need to be determined to evaluate its potential as a drug candidate.

Structure-guided drug discovery (**Figure 1**) is an integral part of most drug design programs and keeps evolving continuously; new technologies are appearing, maturing, and replacing existing ones, thus emerging as new standards. By the early 2000s, high-throughput screening (HTS) became a dominant strategy in the field. This led to an increase in marketed drugs against well-identified molecular targets. However, when examined against new or more complex targets, vast libraries of compounds produced less satisfactory results. At the same time, the colossal size of the chemical space needs to be considered. The number of possible small molecule entities with molecular weights approximating a drug is estimated to be 10^{63} – more than the number of stars in the universe [2]. In this context, a library of ten million compounds at the upper limit of those used in traditional HTS seems trivial. However, alternative approaches, namely, the fragment screening approach, have been developed, addressing some of these issues.

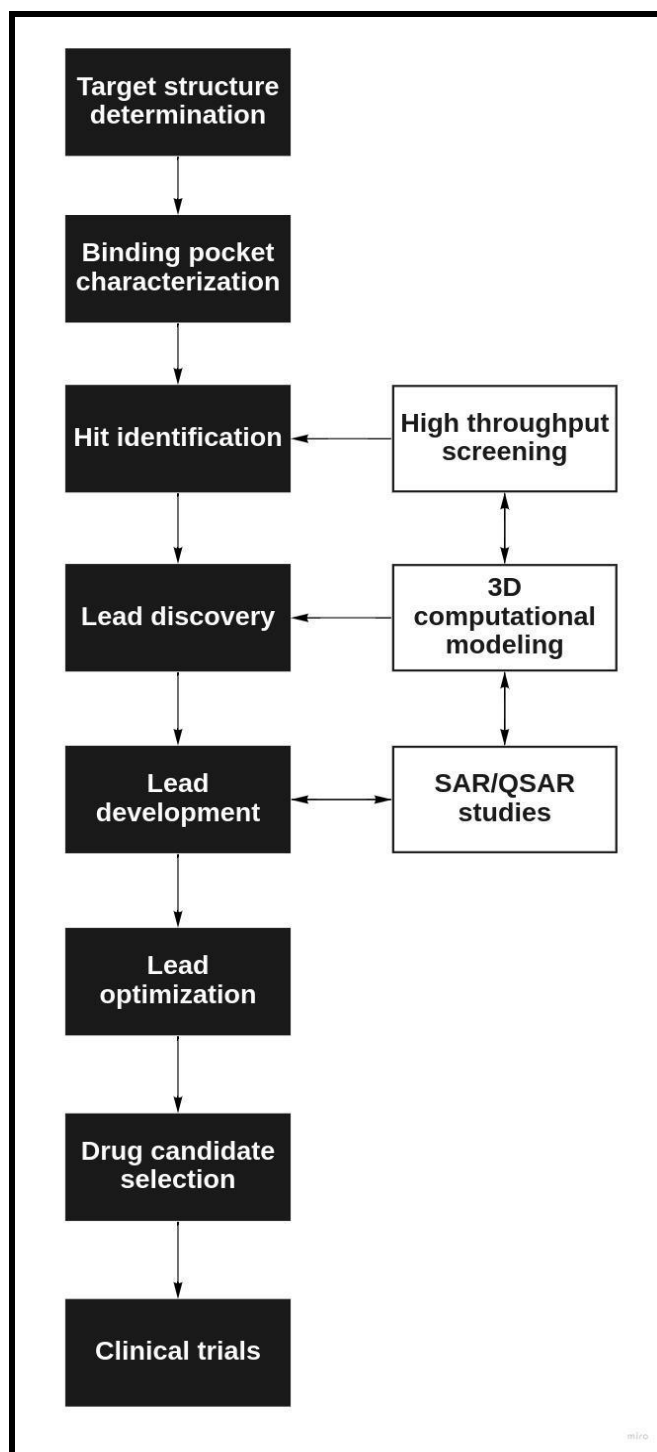


Figure 1 - Schematic of the structure-based drug discovery pipeline [3]. After identification of the drug target, its structure and drug-binding site are determined. Atomically resolved structures enable rapid refinements of lead compounds and optimization of the drug candidate. SAR investigates the relationship between a molecule's biological activity and its three-dimensional structure. The QSAR model is when relationships become quantified [4].

1.2. Identification of chemical leads

1.2.1. High-throughput screening

HTS comprises the screening of large chemical libraries for activity against biological targets via automation, miniaturized assays, and large-scale data analysis. HTS is currently one of the dominant paradigms for lead identification in the pharmaceutical industry. The method screens up to a few million compounds against the target of interest via an activity or interaction assay. Identified hits are then followed up and optimized into chemical leads.

1.2.2. Fragment-based drug discovery

Fragment-Based Drug Discovery (FBDD) identifies small molecules binding to targets. It involves the screening of low-molecular-weight compounds against a target macromolecule of clinical significance. The process helps identify starting hits for the development of potent and bioactive molecules through medicinal chemistry. FBDD emerged as an alternative to HTS. It employs a different approach: Instead of screening millions of compounds to find drug-sized starting points, FBDD starts with smaller pools of smaller compounds. Fragments have less than twenty heavy atoms. Unlike the astronomical number of drug-like molecules with up to thirty heavy atoms, the number of possible fragments (the fragment chemical space) is much smaller. Thus, it can be thoroughly screened with just a few hundred experiments. In the most modern version of fragment screening, libraries composed of 500 to 1000 or more fragments are screened by X-ray crystallography against a target protein. This leads not only to the identification of fragment hits (typically between 2% and 10% of the fragments in the library) but to a complete structural characterization of the three-dimensional binding mode of these fragments, which is then used to facilitate their optimization into potent molecules with drug-like properties. The drawback of this approach compared to traditional HTS is that fragments being small, tend to show weak binding and moderate specificity, also occupying only parts of the active cavity.

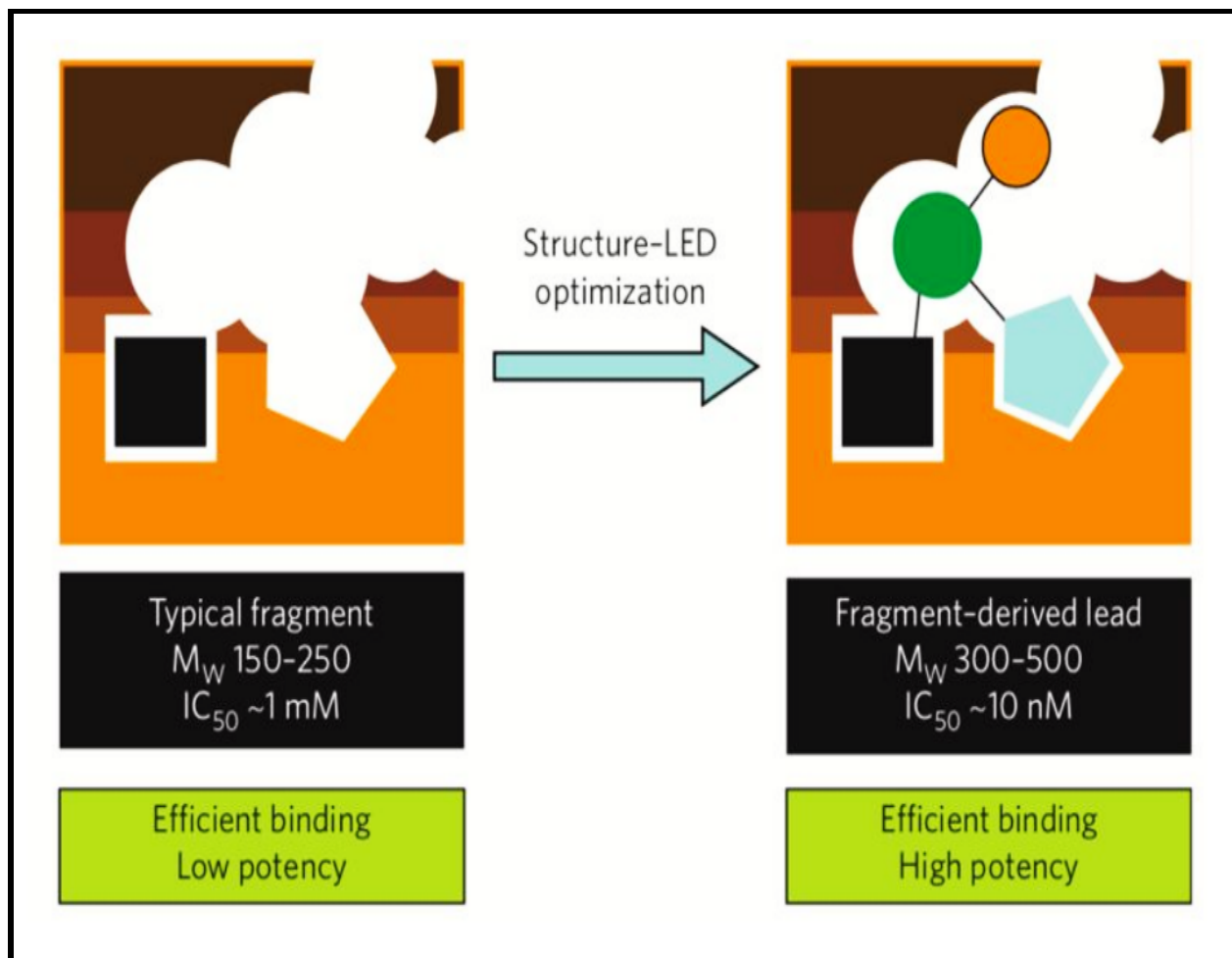


Figure 2 - Fragment-based drug discovery [2]. Around one thousand fragments are screened. The fragments are around 150-250 M_w . The chosen fragment library contains a diverse set of chemical functionalities that are represented in drugs. The binding mechanism of the fragment hit is established in the target protein. Fragments are grown to form new interactions using structure-based drug design. Fragment hits can create high-quality interactions and can be optimized into potent lead molecules, despite their initial low potency due to their small size.

Fragments need to be optimized into bigger compounds with higher affinity and specificity through medicinal chemistry, for which the structural information produced is valuable (**Figure 2**). On the other hand, fragment screening can be applied when HTS fails, for example, when no identifiable activity or binding assay compatible with HTS is available. Another advantage of the fragment screening approach is that it can also help identify allosteric binding sites. Currently, both HTS and fragment screening approaches are routinely applied in drug design.

1.3. X-ray based high-throughput fragment screening

1.3.1. Detecting fragment hits in crystallographic datasets

X-ray crystallography can be used to obtain atomic and molecular structures of biological molecules (**Figure 3**). When a powerful and highly focused X-ray is shone through a protein crystal (typically at a synchrotron facility), the crystalline atoms cause a beam of incident X-rays to diffract into several directions. After measuring the angles and intensities of these diffracted beams and after thorough data processing, the three-dimensional distribution of the density of electrons within the crystal can be generated. This allows determining the mean positions of the atoms in the crystal as well as their chemical bonds and their disorder. X-ray crystallography is commonly used as a screening tool for FBDD. Efficient fragment cocktail design and soaking methodologies have evolved to maximize throughput while minimizing false positives/negatives [5], [6]. Technical improvements at synchrotron beamlines have increased data collection rates. The combination of resources and efficient experimental design has resulted in multiple successful crystallographic screening campaigns. The three-dimensional crystal structure of the fragment bound to its target enables structure-based drug design while revealing insights regarding protein dynamics and function. X-ray crystallography provides structural information that enables rapid and efficient assessment of hits concerning tractability for structure-based drug design and fragment expansion. However, the application of X-ray crystallography as a primary screening method has been under-appreciated due to its relatively low throughput and highly resource-intensive nature [7]. Nevertheless, recent technological developments have changed this [6].

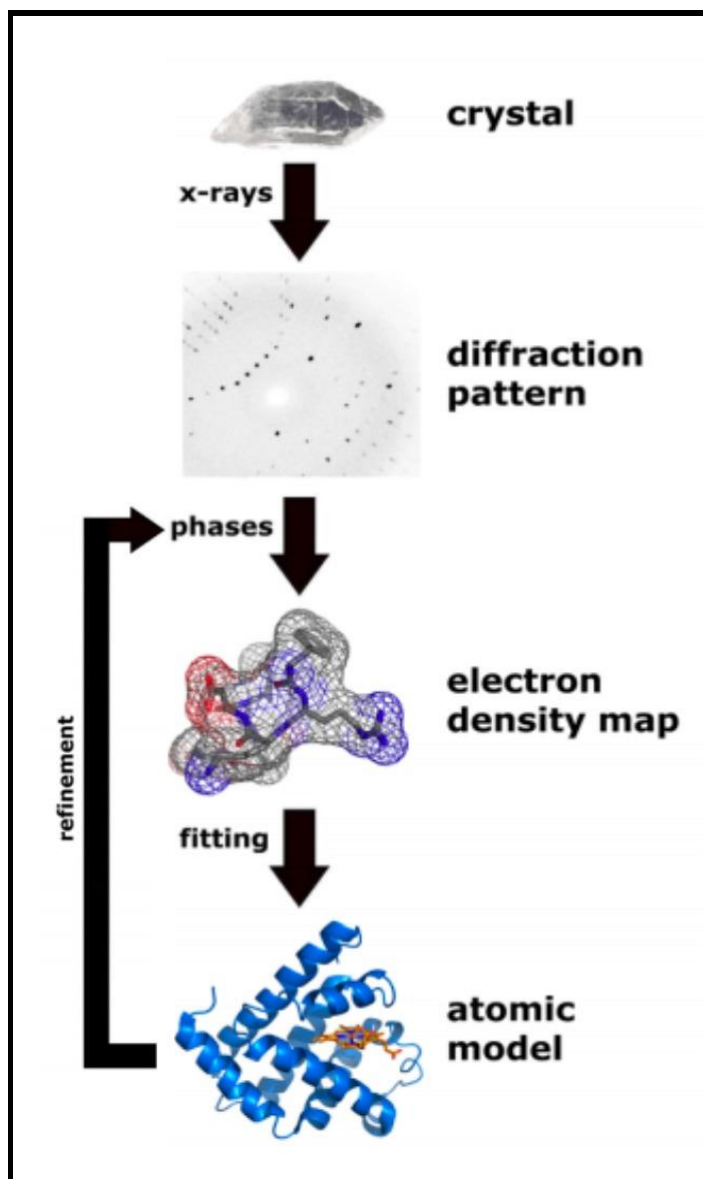


Figure 3 - Workflow for solving molecular structures by X-ray crystallography [8]. First, high throughput crystallization techniques are used for the preparation of crystals with high diffraction power. Crystals should be large enough to produce a regular and high-resolution diffraction pattern when placed in an intense beam of X-rays. Next, the intensities and angles of diffracted X-rays are measured, with each compound having a unique diffraction pattern. Previous reflections disappear, and emerging ones appear along with the rotation of the crystal. At every orientation, the intensity of every spot is recorded. Finally, multiple diffraction images constituting a full dataset are collected and combined computationally. These contain chemical information that enables obtaining and refining a model from the arrangement of atoms within the crystal. The subsequent refined model of the atomic arrangement is called a crystal structure. Any crystallographic structure described in a publication should be deposited in a public repository, namely the PDB.

1.3.2. Exploiting the results of fragment screening for drug development

A fragment screening campaign produces multiple fragment hits, each of them binding to different regions of the same binding pocket. The following step uses this structural information to develop optimized compounds - with high affinity and high selectivity - through molecular modeling and medicinal chemistry (**Figure 2**) [9].

The fundamental principles of FBDD have changed the strategies of drug discovery. Nonetheless, the main technical challenge for fragment discovery is knowing how to optimize fragments into leads. As discussed, structural information has been vital in identifying the optimal vectors for fragment evolution, but faster approaches for compound optimization are still necessary. Most improvements in compound binding affinity, particularly in the early stages of optimization, prevent the protein–compound complex dissociation. On the other hand, the low throughput aspect of X-ray crystallography is now being addressed by laboratories worldwide, such as the high throughput crystallization facility at the EMBL in Grenoble. Thus, contributing to providing more fragment hits for each project and contributing to a more significant impact of X-ray-based fragment screening in drug design (**Figures 4 and 5**).

1.4. Structural data processing, analysis, and integration

1.4.1. Data collection

Diffraction experiments are carried out after obtaining a crystal. First, the X-ray irradiating through the crystal is diffracted and recorded. Then, diffraction data, including the location and intensity of diffraction points, are recorded by specific detectors. A complete dataset (about 3000 diffraction images) can be collected in a few minutes in modern beamlines at synchrotrons. In addition, numerous samples can be tested in a day because of fully automated sample changers and beamlines. Therefore, an entire fragment screening campaign can be completed within just a few days of data collection time.

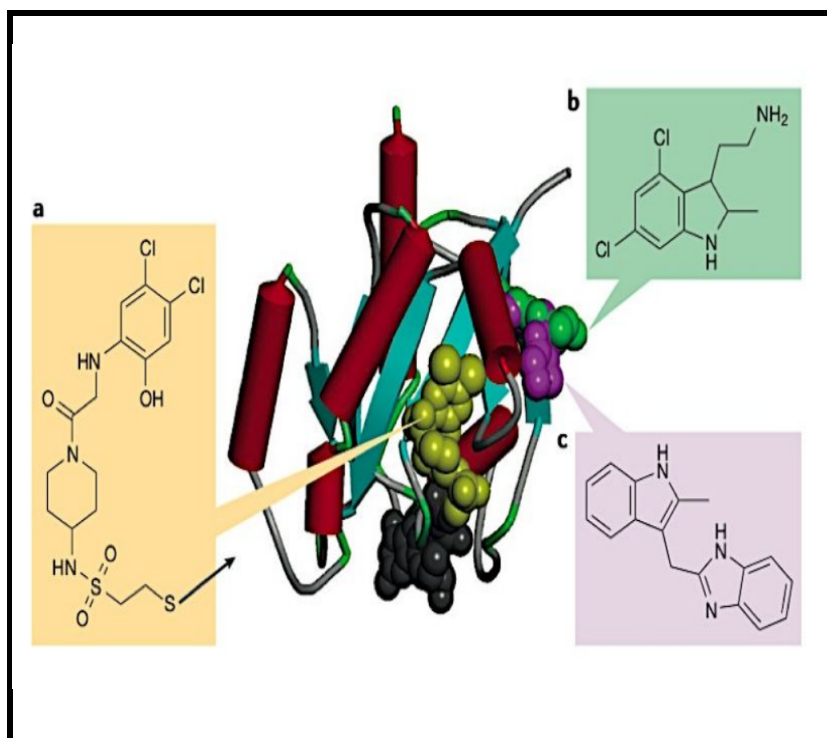


Figure 4 - Fragments binding to the RAS protein [10]. RAS is a GDP/GTP-binding guanosine triphosphatase implicated in signal transduction of cell growth and differentiation. The schematic shows the secondary structure of RAS (α - helices in red and β - sheets in blue) bound to guanosine diphosphate (in grey). (a) A molecule was designed to covalently tether to an oncogenic RAS-G12C mutant (bonding indicated by the arrow). (b) (c) Small fragments bind to similar sites on RAS.

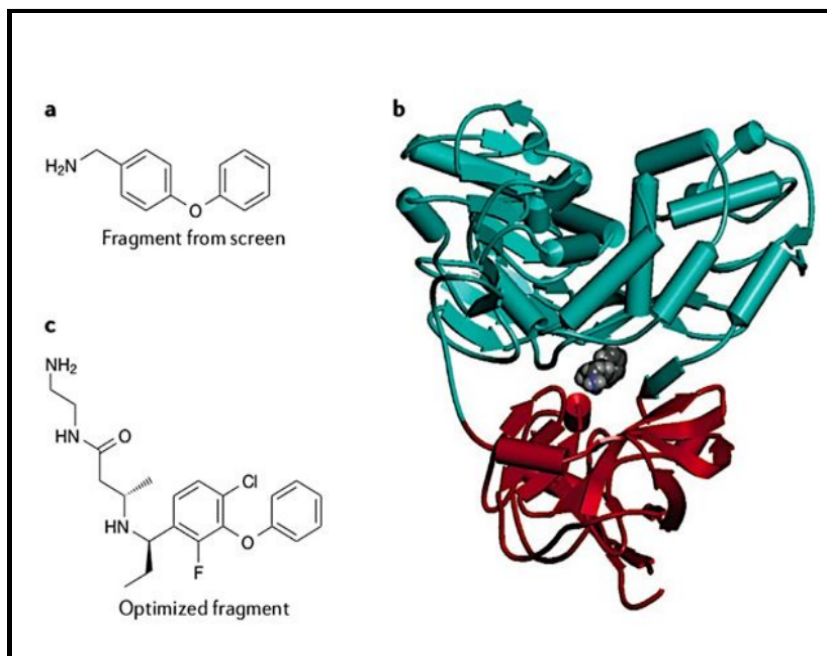


Figure 5 - Discovery of a novel binding site in Hepatitis C virus NS3 [11]. (a) Fragment screening identified a fragment bound in a previously unrecognized cleft of the Hepatitis C virus protease/helicase NS3 (b). That cleft is found between the helicase (blue) and protease (red) domains. (c) In addition, optimization generated a compound that locked the domains together, which leads to activity inhibition.

1.4.2. Data processing

The analysis of the diffraction data allows us to obtain the electron density, which will, in turn, be used to build the atomic model of the protein (**Figure 3**). Pipedream (**Figure 6**) is a fully automated data processing pipeline that facilitates the use and integration of autoPROC [12], BUSTER [13], Rhofit [14], and Phaser [15] into a high-throughput fragment screening pathway. It automates:

- Data reduction and indexing with autoPROC.
- Molecular replacement with Phaser.
- Structure refinement with BUSTER.
- Automated ligand fitting with Rhofit, with subsequent BUSTER post-refinement of the top solution.

The pipeline processes the collected diffraction images to extract indexes and intensity for each of the reflections. These are then scaled and merged to produce a single diffraction dataset. Finally, by applying the phases to the crystallographic dataset (in fragment screening, this is achieved by the molecular replacement method), it is possible to reconstruct the electron density and build the atomic model.

The diffraction experiment data represents a reciprocal space of the crystal lattice [16]. The position of each diffraction 'spot', whose intensity is recorded and proportional to the square of the structure factor amplitude, is governed by the size and shape of the unit cell and the inherent symmetry within the crystal. The structure factor contains information about the amplitude and the phase of a wave. Both of which should be known to obtain an interpretable electron density map that enables building a molecule model. However, the phase cannot be directly recorded during a diffraction experiment. This phenomenon is known as the phase problem. Instead, the initial phase estimates can be obtained in various ways, such as *Ab initio* phasing and molecular replacement. In fragment screening, the method used is the molecular replacement.

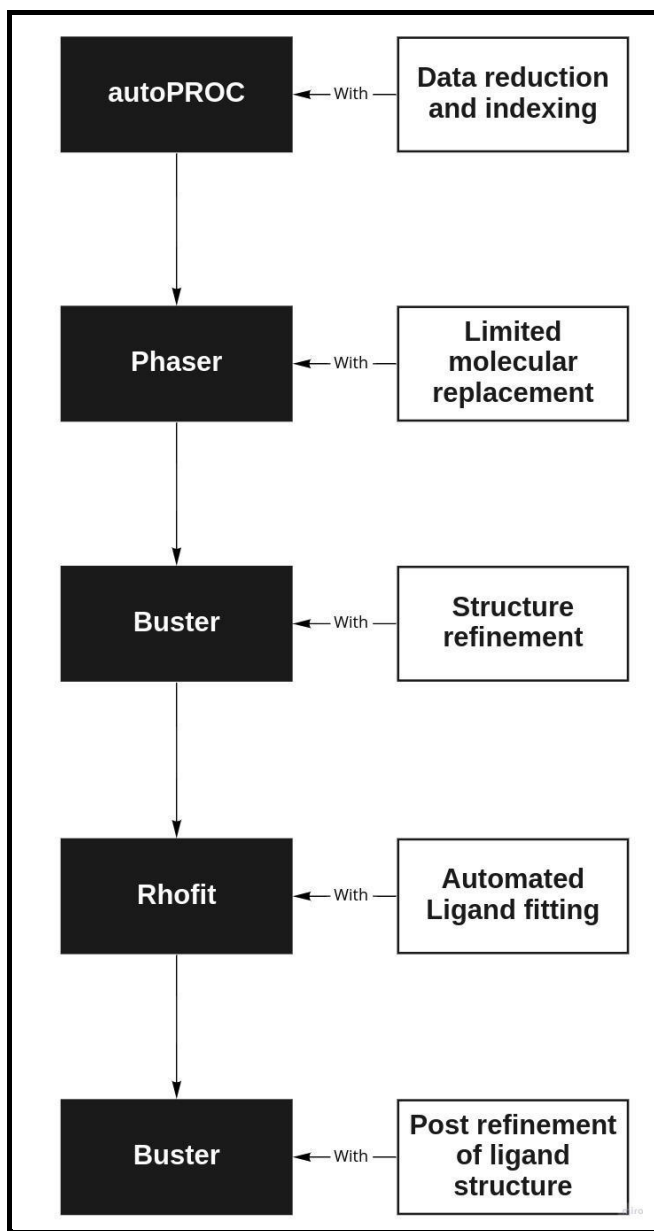


Figure 6 - Schematic of Pipedream's workflow [17]. Pipedream is a pipeline that automates data reduction and indexing with autoPROC, molecular replacement with Phaser, structure refinement with BUSTER, and ligand fitting with Rhofit, with subsequent BUSTER post-refinement the top solution. Additional details about each tool are available in the supplementary data section.

An initial atomic model can be established automatically after obtaining initial phases by molecular refinement. However, this model is typically incomplete or contains errors. This model can be manually improved by evaluating the adjustment of the atomic positions to the electron density, thus getting an improved model and ideally yielding a better set of phases. A new model can then be fitted to a novel electron density map. Finally, further refinement is performed, which continuously proceeds until the correlation between the diffraction data and the model is maximized. Model quality indicators include chemical bonding features of stereochemistry, hydrogen bonding, and the distribution of bond lengths and angles are complementary measures.

1.4.3. Large scale data management in structural biology

The volume and diversity of biological data have increased dramatically. It is estimated that by 2025, structural data, alongside genomics, will generate one zetta-base per year. NoSQL databases have played an essential role in managing these large volumes of data [18]. Omics have recently become a target of the NoSQL movement. NoSQL can be regarded as an umbrella term for non-relational database systems that provide alternative mechanisms for storing, retrieving, and modeling data to traditional relational databases and SQL. There are different types of NoSQL database models such as key-value, wide column, document-oriented, and graph databases. In addition, some NoSQL databases may be hybrids, using more than one database model at once.

Graph databases assemble abstractions of vertices and relationships in connected structures, making it possible to build models that are mapped closer to the real problem. Graphs often represent datasets and their relationships, and the importance of the information embedded in relationships has caused the increase of graph database initiatives [19]. It occurs due to various factors, such as recommending systems, circuits in engineering, social media, chemical, and biological networks [20].

Graph databases are database management systems (DBMS) with create, read, update, delete (CRUD) methods, which can store graphs natively or emulate them in a different database model [21]. A critical aspect of graph databases is that they are highly efficient at capturing relationships, making it possible to discover strong associations between data points in real-time.

1.5. High-throughput fragment screening at EMBL Grenoble

EMBL Grenoble hosts the High Throughput Crystallization (HTX) laboratory, a large-scale user facility providing crystallography assistance. Since 2003, the platform has offered services to over 800 scientists and handled over 1000 samples per year. The laboratory focuses on developing new methods in macromolecular crystallography, including methods for sample evaluation and quality control [22] and the CrystalDirect™ technology. This enables fully automated crystal mounting and processing [23], [5]. The facility has also developed CRIMS (*Figures 7 and 8*), a web-based laboratory information system that automates communication between crystallization and synchrotron data collection facilities. The combination of the CrystalDirect™ technology and CRIMS gave birth to the concept of online crystallography (*Figure 7*). This includes fully automated, remote-controlled crystallography pipelines integrating crystallization screening, crystal optimization, crystal harvesting, crystal mounting, and cryo-cooling and automated X-ray data collection [24] into continuous workflows controlled through web interfaces. This approach accelerates the progression of challenging projects by minimizing the delay between crystal growth and measurement. Moreover, the automation of data collection and data processing in collaboration with GPhL allowed it to build a fully automated, remote-controlled crystallography pipeline to support high-throughput crystallization screening, ultra-rapid structure solution, and large-scale fragment screening in drug design programs. However, improved data processing and analysis pipelines are needed to cope with the volume of data generated.

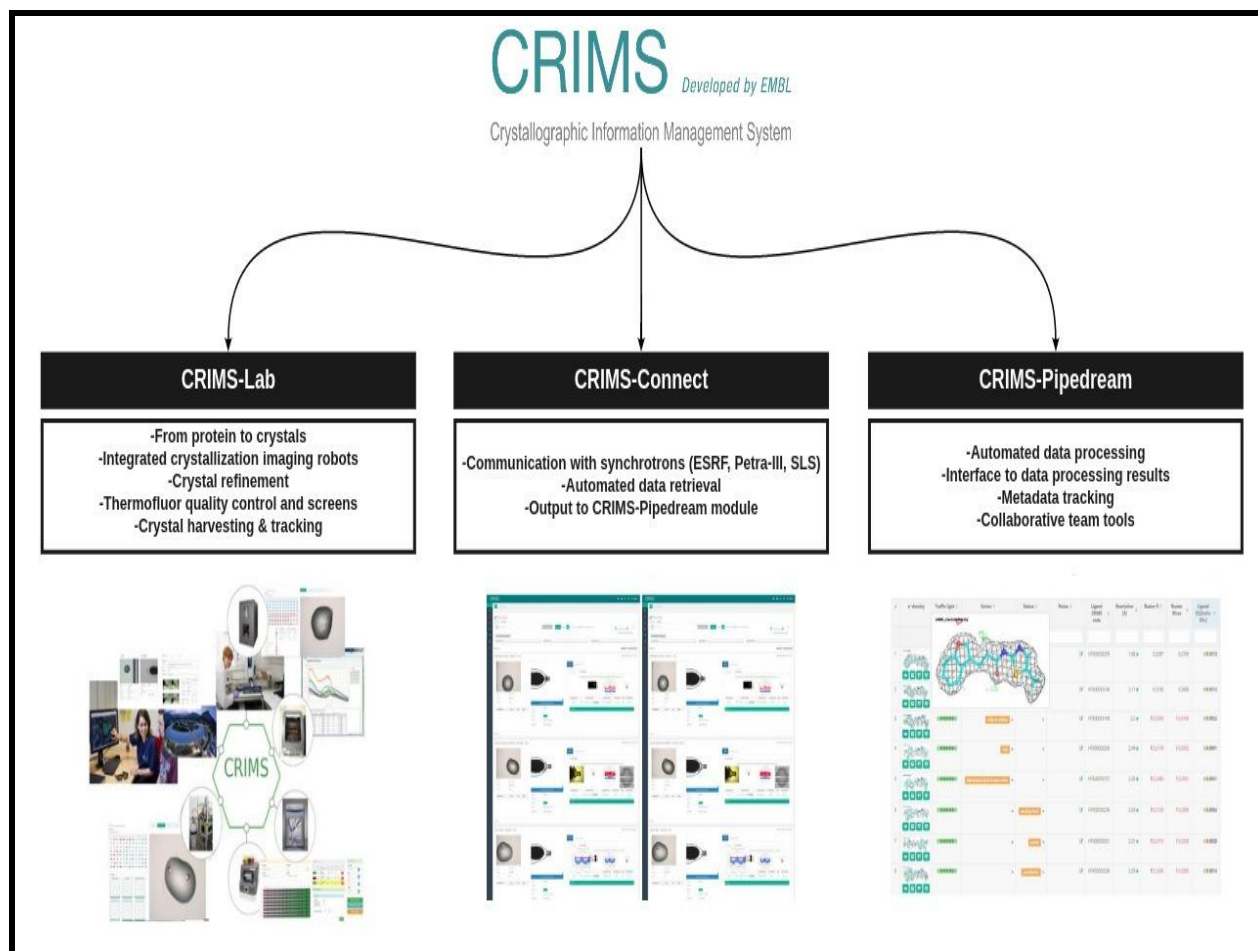


Figure 7 - Schematic of the concept of online crystallography [25]. CRIMS is built upon three modules: **CRIMS-Lab** for high-throughput crystallography experiment design, **CRIMS-Connect** for automated data retrieval, and **CRIMS-Pipedream** (collaboration with GPhL) for automated data processing. This combination speeds up the analysis of compound and fragment screening campaigns and gives real-time access to the results.

1.6. Project outline

New technologies for X-ray-based fragment screening have enabled the screen of extensive fragment libraries: more than 1000 fragments per target and multiple targets per year at a single facility. This opens up new opportunities for developing chemical tools and translational research in the academic field and drug development in the pharmaceutical industry. However, the increase of data generates the need for efficient high-volume data processing. One of the problems in the field is that the characteristics of the different datasets within a single project may vary in quality and properties. However, current data processing pipelines use generic parameters for the whole project. Therefore, they can not adapt to the properties of the specific datasets, which affects the quality of the results and can hinder the identification of fragment hits.

The project aims to introduce novel approaches for data analysis to optimize automated data processing associated with fragment screening at the HTX laboratory in EMBL Grenoble. It involves architecture, design, and implementation of a new web service as part of an effort to refactor and optimize data processing in CRIMS, the software used at the HTX lab, and several other facilities in Europe for crystallographic data management and analysis. The amount of data produced at this and other similar facilities is rapidly increasing. This generates challenges in building efficient data processing pipelines with sufficient capacity to process the data promptly and creates new opportunities as large amounts of data from multiple projects are centralized in a single infrastructure, allowing the introduction of advanced computing and machine learning approaches. The current work focuses on refactoring the existing data processing infrastructure by introducing a new service built upon an underlying graph database. In addition, we approach graph databases to investigate their advantage over relational databases in response to these particular data challenges:

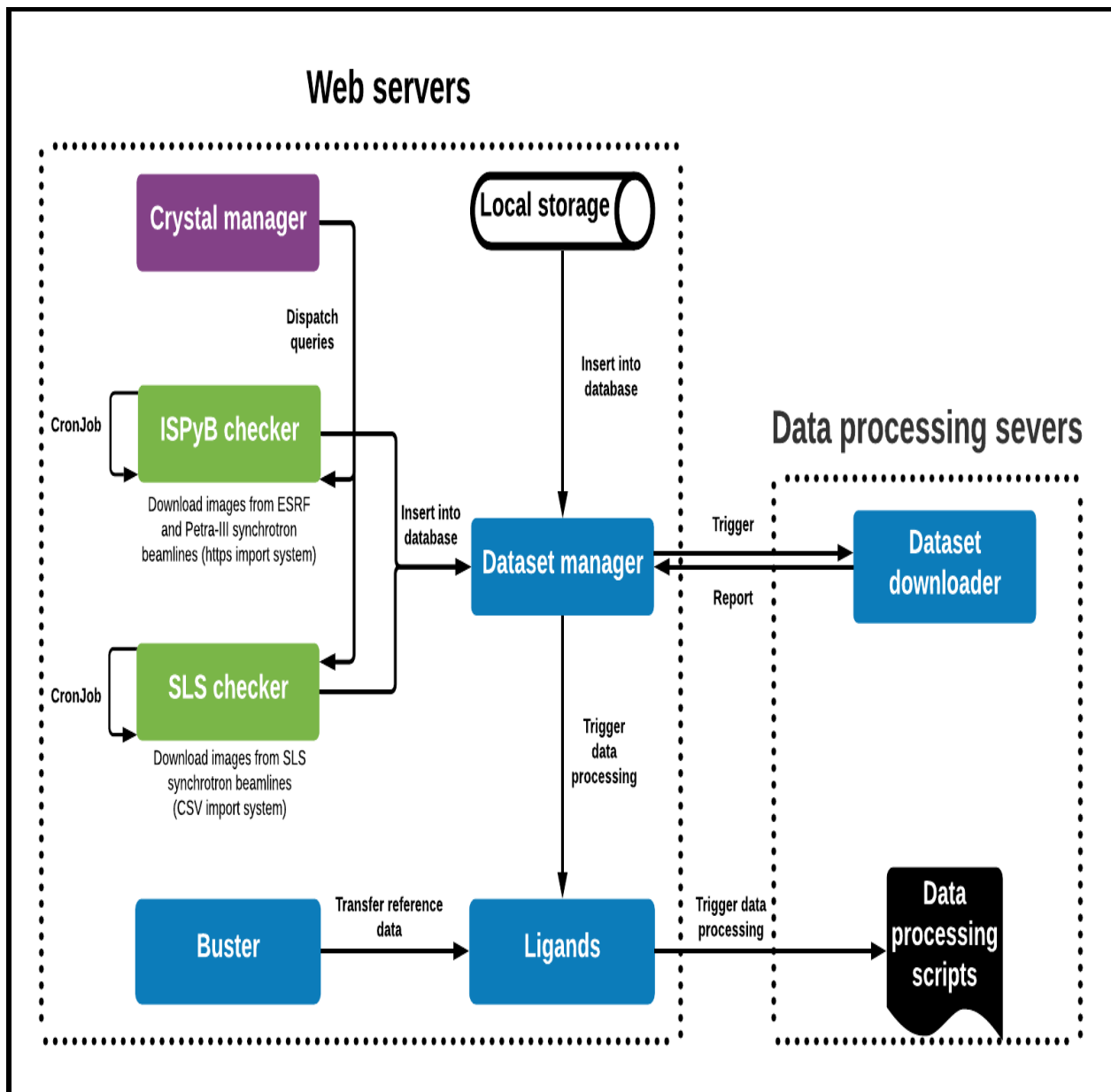


Figure 8 - CRIMS service-oriented architecture (SOA) structural components. The system operates through small and lightweight services that execute specific and single functionalities. The application encompasses seven main services: **crystal manager**, **ISPyB checker** (retrieves data from ESRF and Petra-III synchrotron beamlines), **SLS checker** (retrieves data from the SLS synchrotron beamlines), **buster** (data processing results and reference storage), **ligands** (data processing specifications and ligands related data), **dataset manager** and **dataset downloader**. These services communicate to provide full functionalities of CRIMS.

- Frequent schema exchanges and adaptability of the data model
- Capacity to deal with the introduction of significant volumes of data.
- Data valence for future artificial intelligence applications, as graph databases focus on relationships between data. New relationships can be easily explored and implemented without the need for migrations.

The performance of a web application relies on a lot of factors. Relational databases impact that performance:

- Changes to the data model are difficult to implement (require migrations).
- Efficient and suitable for integrity. However, new and unexpected relationships between data may be difficult to mine and extract.
- Whenever multiple joins are needed, performance decreases.

Refactoring data processing in CRIMS is expected to be more critical with the increased volumes of data generated, making performance a key factor. A faster loading web service gets better user engagement. The speed optimization influences conversion and usability of CRIMS. More efficient data processing improves user experience (UX) [26], leading to higher productivity and impacting translational research and drug design. Implementing a novel graph database prototype establishes the general concepts and techniques that will allow the expansion and modification of CRIMS data processing resources to meet new challenges in X-ray-based data processing and analysis. The new prototype web service is expected to offer an initial framework into which various other data sources, including additional data processing software, could be incorporated.

1.7. Project development strategy

1.7.1. Brainstorming

The first step was to elaborate a development strategy (**Figure 9**). We had to decide goals and objectives, make the appropriate analysis associated with refactoring data processing in CRIMS, and formulate a list of tasks. These steps were driven by input from ALPX and GPhL, as they are in direct contact with users. Additional information from the HTX facility members was taken into account. It included feedback on organizing data processing output to ease the subsequent data analysis.

1.7.2. Planning and research

The next step was developing scheduled work and choosing the technologies and tools to build suitable infrastructure services. We had to analyze the collected requirements and create agile user stories [27] on that basis. This provided a general explanation of the novel data processing service features written from the end-user's perspective and articulated how a feature will give value to the user. This ensured a user-oriented development process, enabled collaboration, and drove creative solutions.

1.7.3. Design and prototyping

This phase involved the backend development of the novel data processing service. First, we started implementing and prototyping the graph data model. Backend development includes installing and configuring the content management systems, database modeling, and frameworks. After completing all the steps in the strategy and design phase, a preliminary prototype was framed and acted as a setup for subsequent agile development.

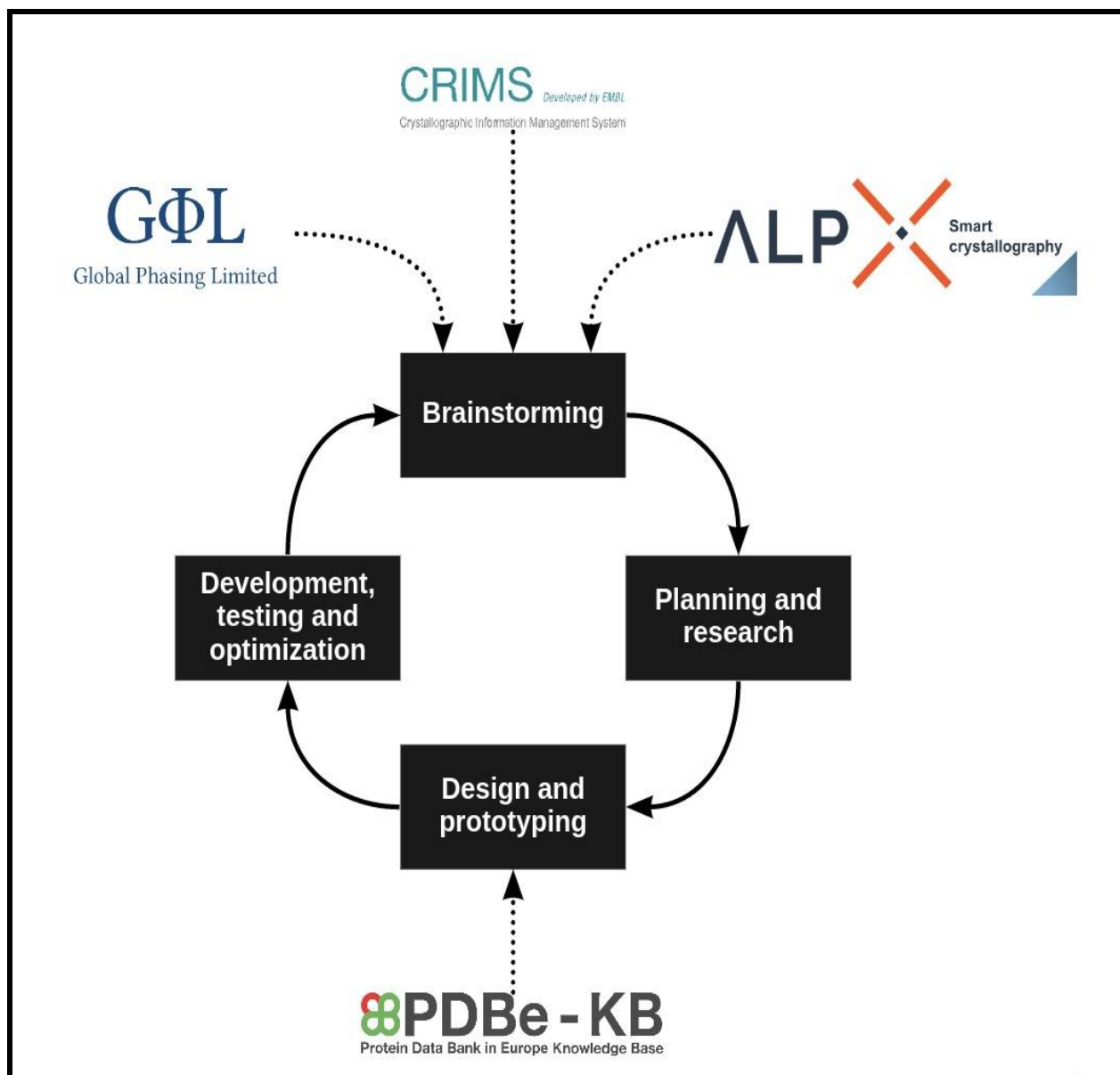


Figure 9 - Schematic of the project's development strategy. Input from different stakeholders helped shape the project's strategy on different levels. Inputs from the ALPX, GPhL, and HTX teams were the main pillars of the project. Their insights were later translated into agile user stories, paving the way for setting up tasks to develop its features. Input from the PDBe-KB team provided insights on efficient graph data modeling and implementation with Neo4j.

1.7.4. Development, data-driven testing, and optimization

Testing was an essential phase in the development process. It was performed to ensure the user's requirements after the implementation of the features. Current testing strategies were data-driven. Representative data was transferred to emulate real-world scenarios. The features of the written code were thus evaluated and validated, and actions for removing bugs were taken.

2. Materials and methods

2.1. Architecture pattern

A set of factors drove the choice of the architecture pattern (*Figure 10*) [28]. These include infrastructure support, developer skillset, project deadlines, and application size. The SOA style structures CRIMS as a collection of services that are: highly maintainable and testable, loosely coupled, independently deployable, and organized around business capabilities. The pattern was chosen for the following reasons:

- CRIMS already implements an SOA pattern. It enables rapid, frequent, and reliable delivery of large, complex applications such as the present. It also allows an organization to evolve the technology stack.
- The pattern allows easier deployment through a practical and streamlined delivery pipeline and a high degree of application and component decoupling within the application.

2.2. Pattern topology

The API REST-based topology [28] was helpful in exposing small, self-contained individual services through APIs. APIs are interfaces that allow machine-to-machine communication. They helped solve problems by reducing the time and cost of developing the service and increased the likelihood of producing effective software.

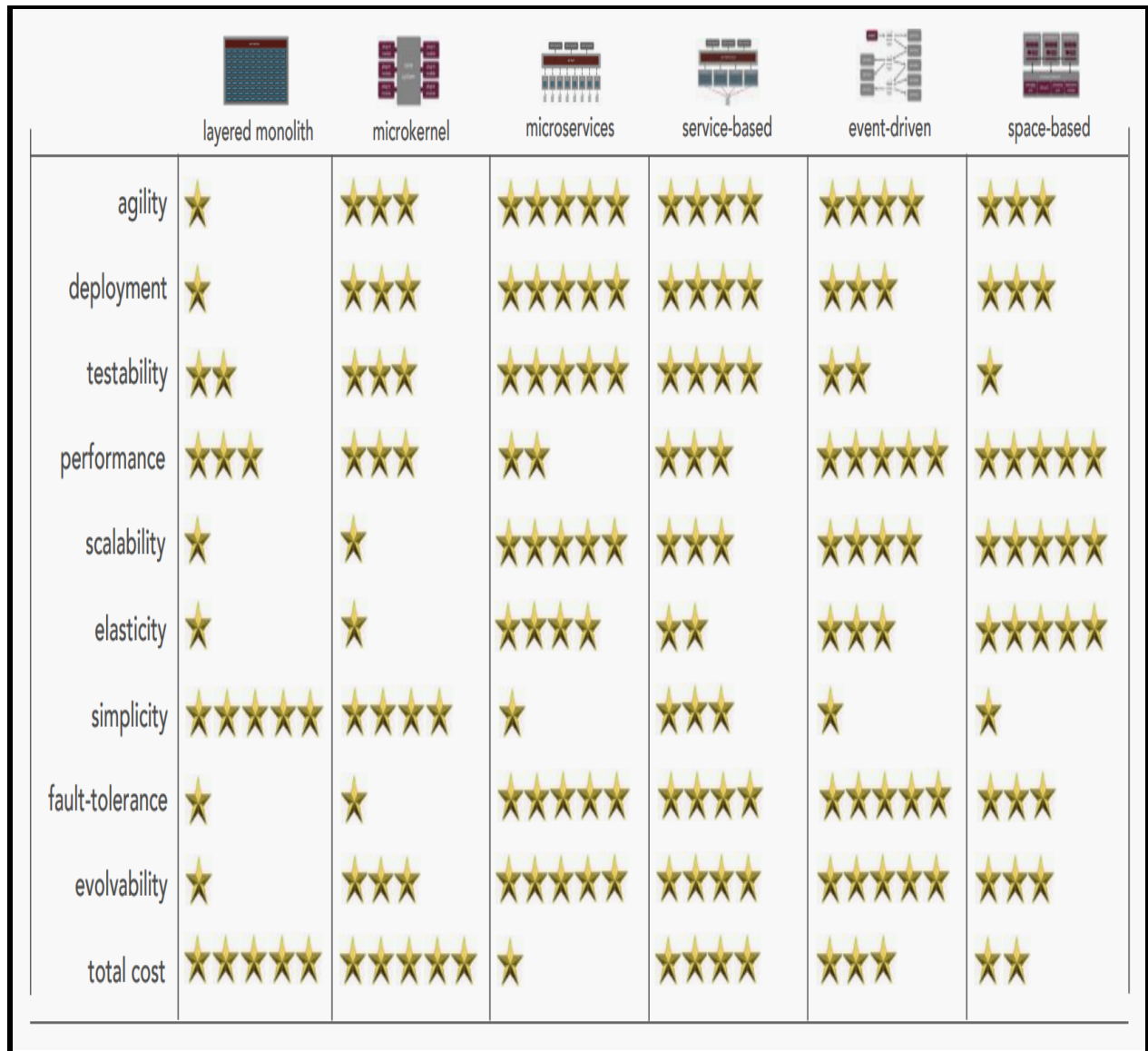


Figure 10 - Trade-offs of different software architecture patterns [29]. The six most used patterns (columns) are compared according to ten features (rows) on a star scale from 1 to 5 (1 is a poor resource and 5 is an excellent resource).

An API is a set of instructions on how two software can communicate. This topology consists of service components that contain modules that perform specific functions independent from the rest of the services. These service components are accessed using a REST-based interface implemented through a separately deployed web-based API layer.

2.3. Framework

2.3.1. Backend framework

Django [30] enabled the rapid development of a secure and maintainable service. Three factors mainly drove the choice of the web framework:

- Database driver maintenance and performance. To build an application, we want to connect to Neo4j from our technology stack. Fortunately, it is straightforward to use a driver which connects to Neo4j via Bolt or HTTP.
- The framework was chosen based on the libraries it provides: Python reduces development time, no need to reinvent the wheel.
- Skillset and available expertise inside the team.

2.3.2. Database management system

The novel data processing service was built upon the freely available graph database Neo4j [31]. Like many other graph databases, it is built upon the property graph model. Labeled nodes are connected via directed, typed relationships. Both nodes and relationships hold arbitrary key-value pairs.

The graph data model was created using Arrows [32], a web-based tool for drawing pictures of graphs for use in documents or presentations. It provides methods to draw nodes, relationships, properties, and labels. In addition, the model can be exported into multiple formats, such as Cypher. This makes it easier to create graphs in a Neo4j database.

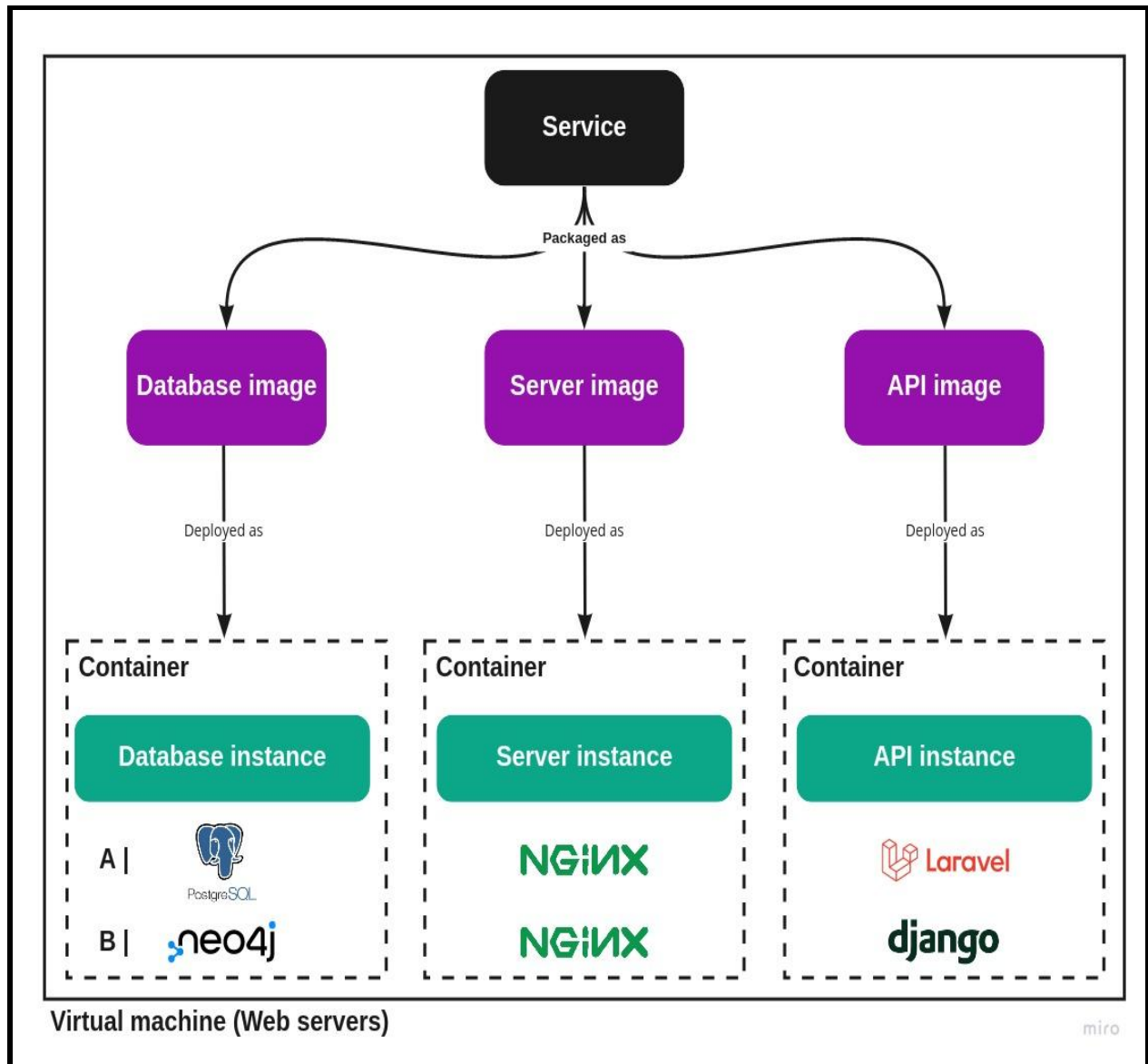


Figure 11 - Underlying microservices of each CRIMS service. Each CRIMS service is packaged as three Docker images: a database image, a server image, and an API image. Each image is deployed as a microservice container instance. **A** | Currently, each service is built upon three microservices: a PostgreSQL [33] database, an NGINX [34] web server - that can be used as a load balancer, and an HTTP cache -, and a Laravel [35] API. **B** | The novel data processing service is built upon a Neo4j database, an NGINX web server, and Django API.

2.3.3. Data accessibility and representation

The database should be accessed through a technology stack. This is implemented using a driver which connects to Neo4j via Bolt or HTTP. Neomodel [36] is an Object Graph Mapping (OGM) for the Neo4j graph database. It enables mapping nodes and relationships in the graph to objects and references in a domain model. Neomodel includes standard Django model style definitions and enforces the schema through cardinality restrictions.

The nodes that are fetched by the API are just instances of the defined model classes. We want to return JavaScript Object notation (JSON) files to the fetch calls made in the frontend. We had to create our serialization methods and manage the logic of those since sometimes we would need to serialize only the directly related nodes to a specific node. Data serialization is the process of getting data from the object format to a parseable format.

The Django REST framework [37] is a powerful and flexible toolkit for building web APIs. It implements:

- Serialization that supports non-ORM data sources.
- Authentication policies, including packages for OAuth1a and OAuth2.
- Customizable, maintained, and extensively documented.

2.3.4. Containerization and deployment

Three Docker [38] images are available with all libraries and dependencies (Neo4j, Nginx, and Django). It allows consistent development and deployment of the code on an extensive range of systems. Docker is used to quickly and confidently deploy working services and to scale horizontally. Each CRIMS service is packaged as three Docker images (**Figure 11**): a database image, a server image, and an API image. Each image is deployed as a microservice container instance.

3. Results

3.1. CRIMS novel SOA

As shown in **Figure 8**, the existing “ligands service” database was built upon PostgreSQL. It stores data related to data processing results/specifications and ligands at once. This initial framework is the main target of the refactoring efforts. The aim of the novel data processing service implementation is two-fold:

- It is meant to replace the “ligands service”: the database layer contains data processing results/specifications without any data related to ligands.
- It sets the premises to the parallel implementation of another new service: “Data processing scheduler.” The latter is expected to emulate a bridge between the web servers and the data processing servers (**Figure 12**). First, all the information needed is transferred from the novel data processing service and the dataset manager service. Subsequently, data processing is then triggered through the scripts stored on the data processing servers.

3.2. Graph data model construction

Amid the brainstorming sessions, we analyzed the collected requirements and created several agile user stories (**Figure 13**). This provided means for expressing a user-centered view of the service's needs and the questions that arise in satisfying this need.

Suppose we take the example of the user story in **Figure 13E**. This story expresses a user's need, which motivates the shape and content of our data model. From a data modeling perspective, the “AS A” clause establishes a context comprising two entities -a user and a data processing step-. The “I WANT” clause then poses a question: which data processing step has the user launched and wishes to retrieve its results?

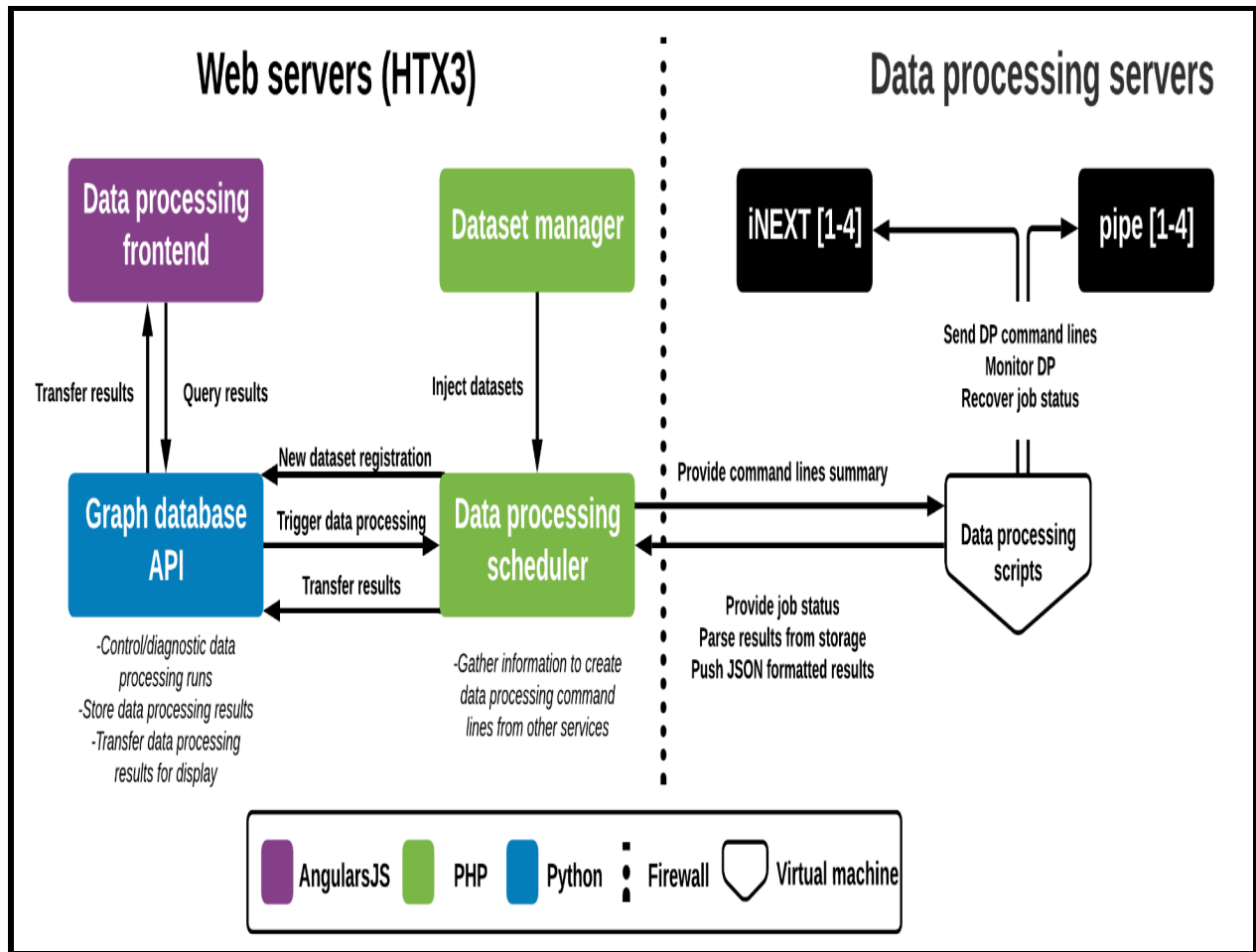


Figure 12 - CRIMS SOA new structural components. Two new services are expected to be added to the system's service stack. First, the novel data processing service (Graph database API) would store everything related to data processing. Second, the "Data processing scheduler" would emulate a bridge between the web and data processing servers. The service follows the same architecture presented in **Figure 11**.

This data model is a direct implementation of the question presented by the user story. Thus, it allows queries that similarly reflect the structure of the question we want to ask of the data. This helps identify entities and relationships in the produced model (**Figure 15**).

The Velankar team manages the PDBe-KB (<https://pdbe-kb.org>), a community-driven, collaborative resource for literature-derived, manually curated, and computationally predicted structural and functional annotations of macromolecular structure data, contained in the PDB. Considering that Neo4j powers PDBe-KB, we fostered the collaboration between our team and the data bank team to get insights and benefit from their expertise in graph database implementation.

The PDBe-KB team expressed that they ran into performance issues with graph databases under certain specific conditions. They had to implement several caching mechanisms to improve performance. The problem was usually seen when they traversed multiple nodes to retrieve data rather than having many properties on a single node. They collected data as it was sensible to view it in a graph rather than ensuring it was organized, so querying was fast.

A case example was given as follows. When they wanted to get interactions between amino acids in a complex, it was quicker to run software on a PDB entry than to get the same data from the graph database. It involved joining lots of nodes: the resulting query is about twenty lines long. Thus, the team is now experimenting with extracting data from the graph database and loading it into a relational database table to power their APIs. The trick here is to use the graph database to collate the data and then provide it in a flat format for the relational database to load in a single table.

That said, the PDBe-KB team's input helped shape our graph data model to improve the end-performance of the novel data processing service. Two crucial points highly impacted the modeling process:

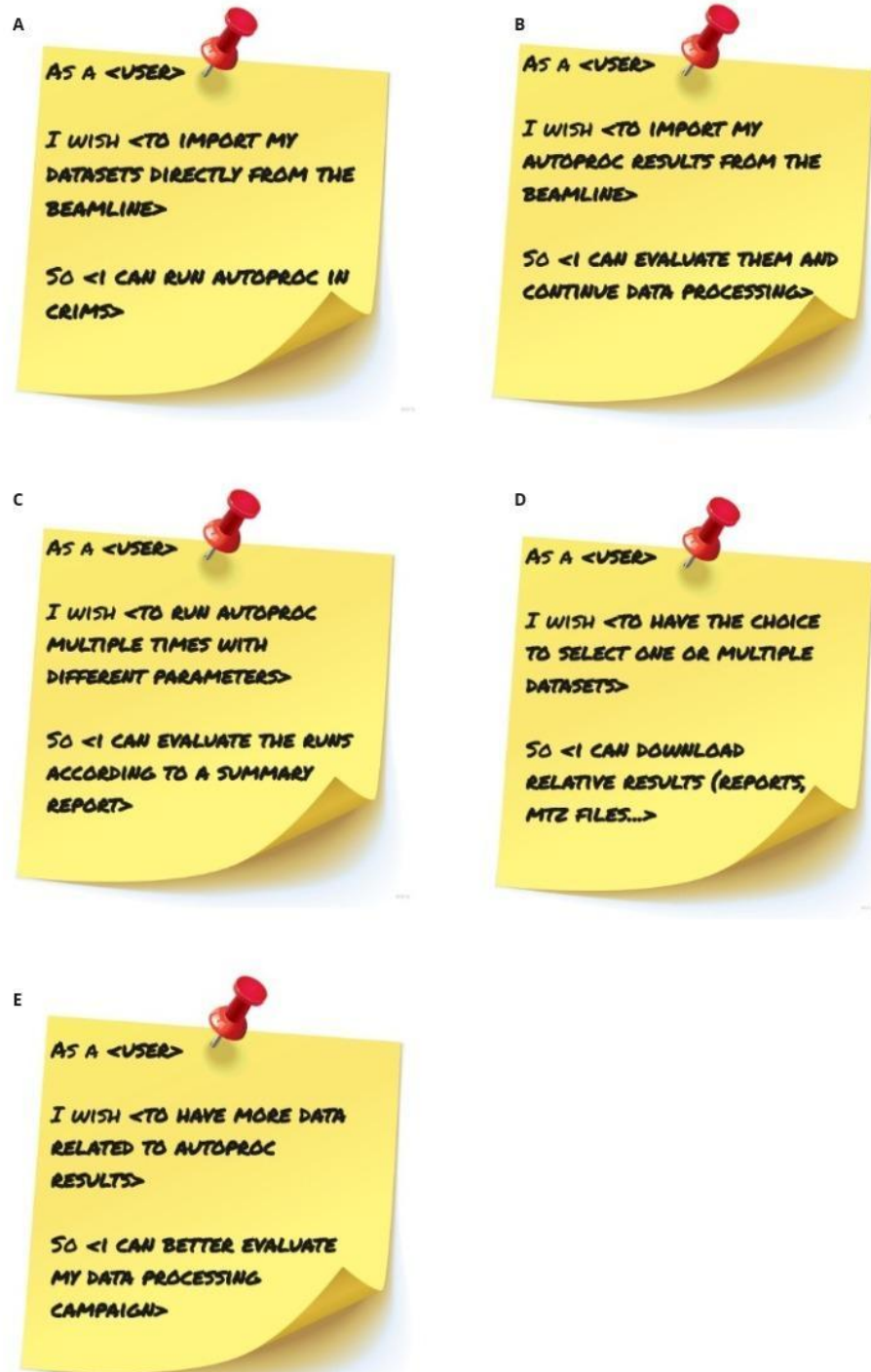


Figure 13 - Agile user stories collected from stakeholders. User needs and requirements generate agile user stories. These are epic stories: they contain an abstract idea. Nevertheless, these stories paved the way for setting up tasks to implement the business logic of the novel data processing service. Equally, API implementation and graph data modeling were driven by these stories. User stories were identified together with the stakeholders through face-to-face meetings.

- Graph databases are better than relational databases for data mining. In the case of the PBDe-KB team, they allowed collating data in ways that were just not possible with relational databases.
- Graph databases are inadequate when querying multiple nodes and properties. Relational databases are better when the data can be fit in well-formatted tables.

Based on this input, we changed the graph data modeling methods in a way that optimized results. Whenever the transition from node properties to separate nodes is possible, the changes were carried out. Furthermore, the development process is heading towards using a hybrid data model: the graph database for data mining and the relational database for collating the collected data in a flat-formatted table to speed up data retrieval and visualization.

The Velankar team's input was reflected in the early decisions we encountered, such as whether to model data as a property on a node or as a relationship to a separate node. For example, the data in **Figure 14A** models coordinate type as a property on the coordinate node. To write a query, finding the type of coordinates is very simple. It would find the coordinate node it wants to know about and then return the type property values. However, to find out which coordinates share types, we would need a much more complex query to find each coordinate node, loop through each of the types in the property array, and compare each value in the second coordinates' property array of types. This would impact performance (nested looping and comparison of node properties), and the query would be much more complex, as well. The code block in **Figure 14B** is what the syntax would look like for each query. A shift in logic and complexity of the loop in the second query appears.

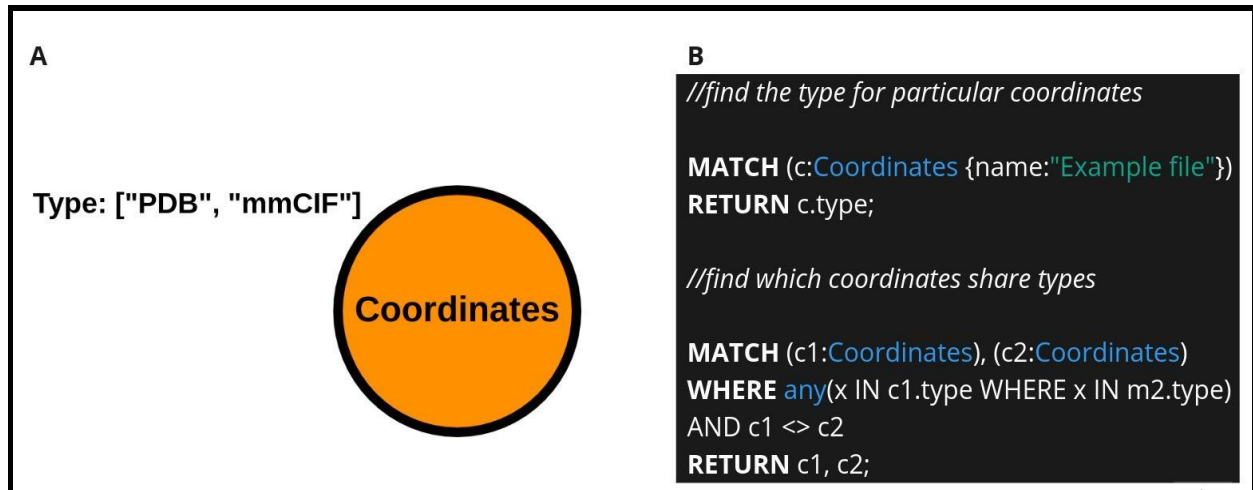


Figure 14 - Modelling data as a property on a node. **A:** the “coordinates” node has one property key with two possible values: PDB and mmCIF. **B:** The first query enables you to find the type for particular coordinates. The second query enables loop through “coordinates” nodes to find those that share the same type.

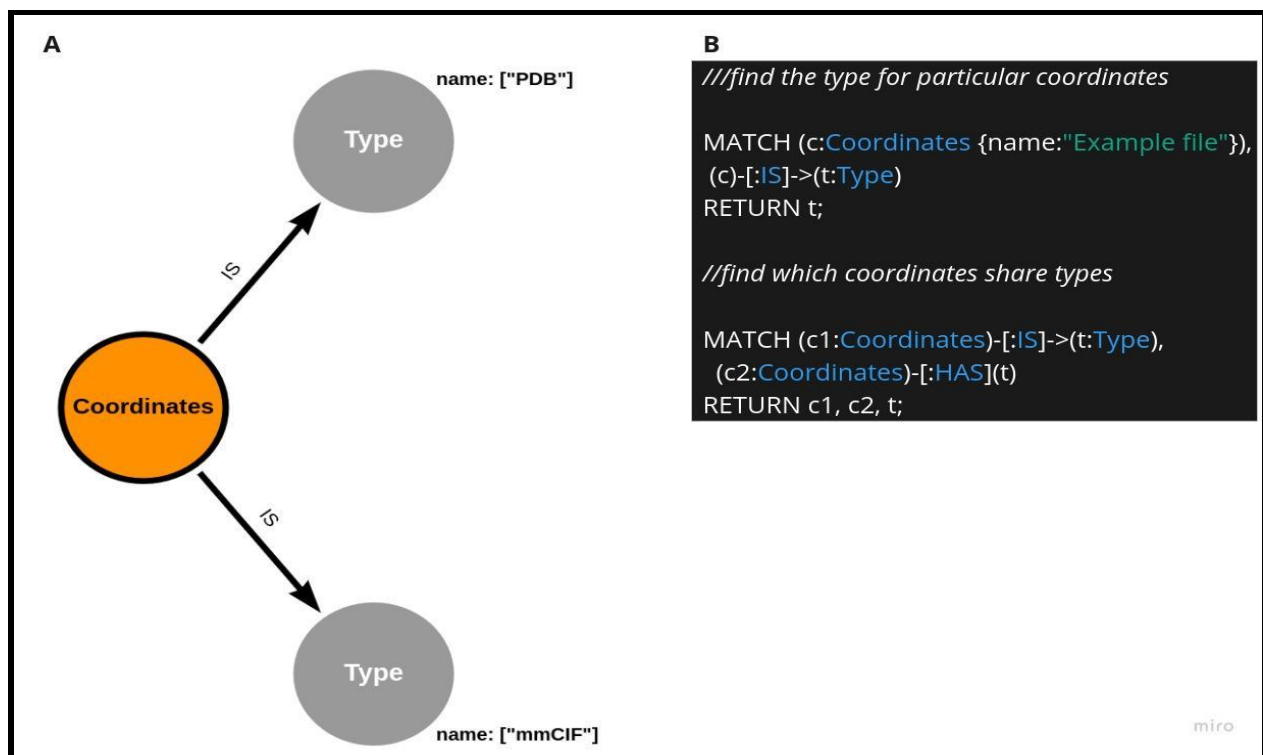


Figure 15 - Modelling data as separate nodes. **A:** the coordinates and types are modelled as separate nodes. **B:** The first query enables you to find the type for particular coordinates. The second query finds the coordinates and the type they are related to, then looks for other coordinates that have the same type.

Instead, if we were to model our coordinates and types as separate nodes and create a relationship between the two, we would develop a model that looks something like **Figure 15A**. This creates an entirely separate node for the type, allowing the connection of all the coordinates with a shared type to that type node. This would implicate changes in the queries. To find the types of particular coordinates, it first needs to find the coordinate node it is looking for (in this case, 'Example file') and then find the node connected to that file through the "IS" relationship. The difference is in the syntax (**Figure 15B**) for the second query to find which coordinates share types. It is much simpler than the earlier version because it uses a natural graph pattern (entity-relationship-entity) to find the information needed. First, the query finds the coordinates and the type they are related to then looks for the second file in that same type.

Both versions of the data model are helpful. However, the optimal option depends on the types of queries we intend to run against our data. For example, if we plan to analyze individual items and return only details about that entity (like types on particular coordinates), then the first data model would serve perfectly well for our needs. However, if we need to identify common ground between entities or look at a group of nodes, then the second data model would improve the performance of those types of queries. The end goal was to enable future use of the novel data processing service in data-mining approaches. Thus, we have implemented the second method in the resulting data model (**Figure 16**).

Agile user stories, input from developers, and data helped generate the latest version of the graph data model shown in **Figure 16** and described in **Figure 17**. It encompasses 39 node types, eight relationship types, and approximately 250 property keys. In addition, the database layer enables the storage of:

- Reflection data (MTZ files) and macromolecular coordinates (PDB and mmCIF files).
- Reports generated from the Pipedream pipeline.
- Information about computation and storage hosts.
- Information about incoming datasets and relative data collection locations and parameters.
- The type of each data processing step and the tool that enabled the step.
- Command-line options for each data processing software.

3.3. Data access layer implementation

Based on the data model mentioned above, the data access layer's components were implemented. REST services for the node and relationship models were implemented using HTTP, which defines four request CRUD methods::

- **GET:** fetch the resource addressed by the URL.
- **POST:** send data to the URL and receive a response.
- **PUT/PATCH:** send a resource to be placed at the URL specified address.
- **DELETE:** discard the resource addressed by the URL.

As shown in **Figure 18**, the first set of instructions would enable the reception and registration of data processing results. The second set of instructions would enable the recovery of data processing results for the frontend display. The third set of instructions would enable the recovery of data processing specifications and settings.

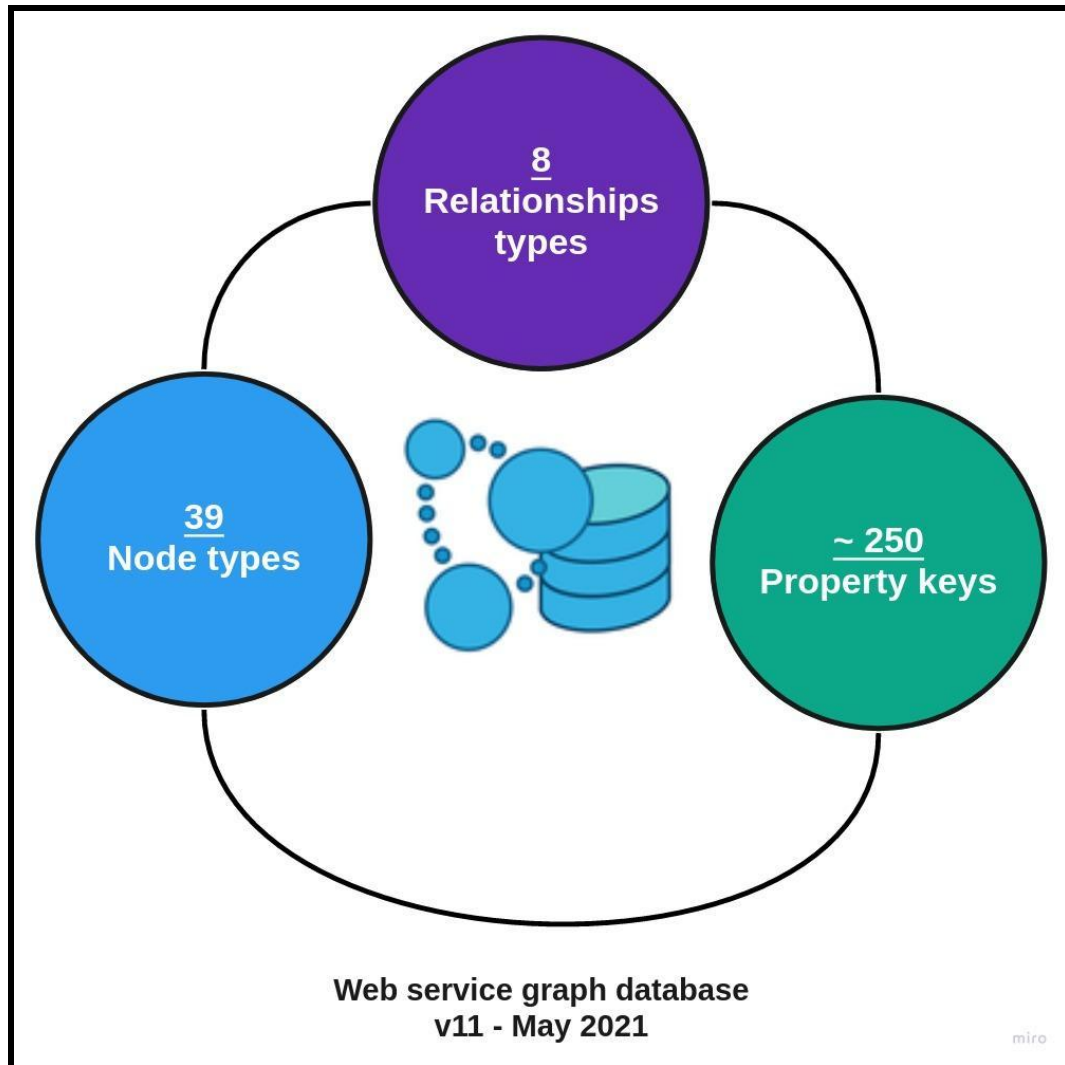


Figure 17 - Database layer specifications of the novel data processing service. The layer is powered by Neo4j (<https://neo4j.com>). The current version encompasses 39 node types, eight relationship types, and approximately 250 property keys.

Data-driven testing (**Figure 19**) using a sample JSON formatted data was carried out to test the implemented instructions. Suppose we take the example of registering data processing settings. The file contains information that would be directly transferred to novel data processing service API gateways. JSON data would then be parsed and stored inside the database layer. The features of the written code were thus evaluated and validated, and actions for removing bugs were taken. We successfully registered data processing settings related to datasets, users, storage host, computation host, data collection, and construct in the given example. When a POST request was prompted to the “/storeInput” URL with the sample data (**Figure 19A**), nodes and properties were created accordingly inside the database layer (**Figure 19B**).

4. Discussions

Since the first implementation of the service, the data model shifted and will keep shifting to meet new requirements. Rigid schemas and complex modeling processes imposed by relational databases portray them as a bad fit for the rate at which data changes over time (data velocity) in this application. To maintain the integrity of the data, the solution required a data model that did not sacrifice performance and supported ongoing evolution. The relational database is considered an adequate tool if the application is well-understood with minimal data model changes. In the case of refactoring data processing in CRIMS, the path led somewhere else. We were trail-blazing into uncharted territory. In the beginning, we could not have established a plan for a database with all the correct answers because user requirements evolved alongside CRIMS. We needed a data model that's agile and grows alongside development without lagging.

The capacity to deal with data volume is another critical driver behind CRIMS adoption of the graph database. In particular, query execution times increased as the size of tables and the number of JOINS grew. This is not always the defect of the relational databases themselves, however. Instead, it has to do with the underlying data model. The graph database came in great help to avoid JOIN pain as it puts relationships at the center [39].

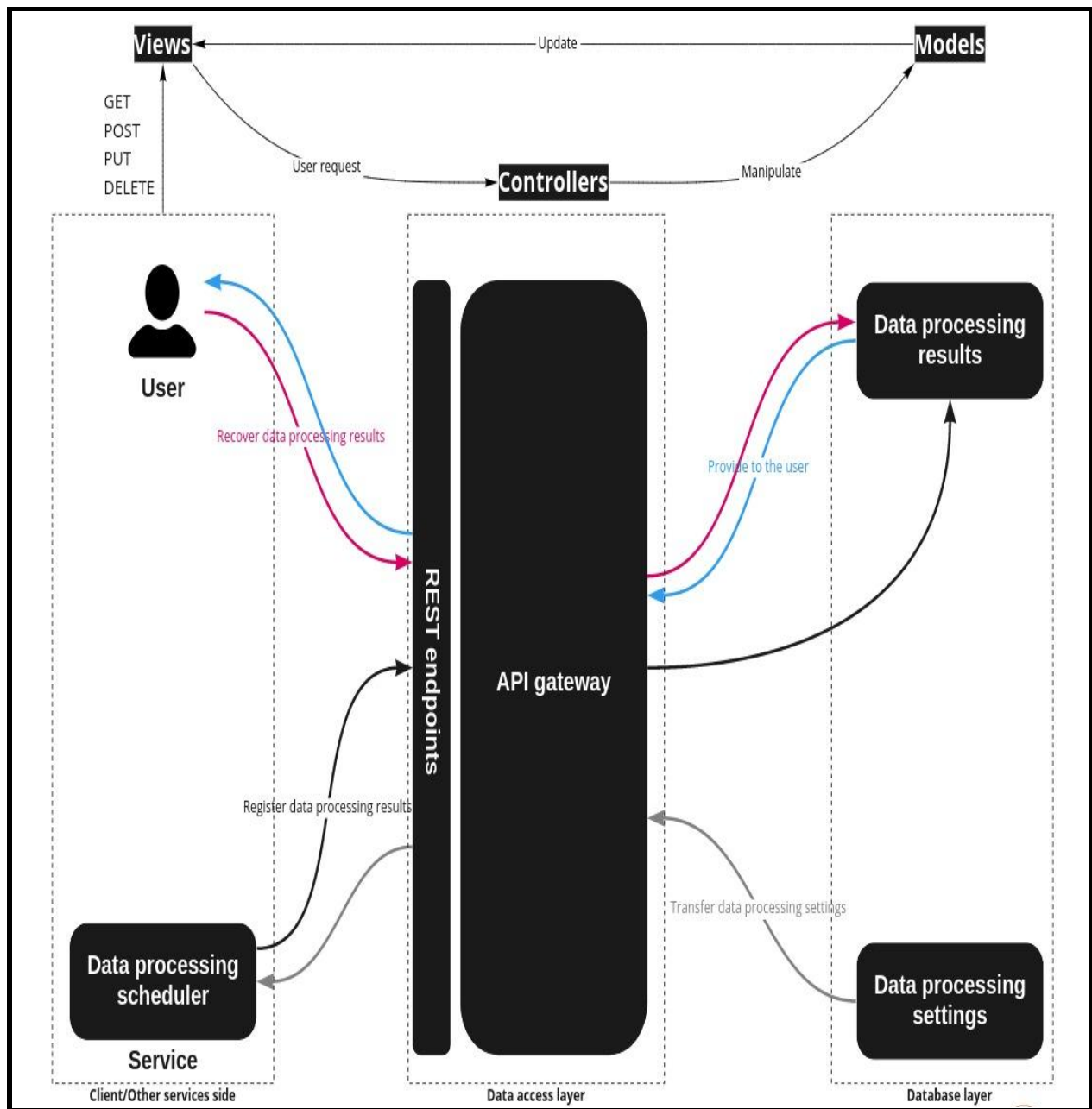


Figure 18 - Layers and interactions of the novel data processing service. An API gateway gives a single entry point and provides an interface to access data, logic, or functionality from back-end services [40]. For example, the current version of the novel data processing service operates through an API gateway. The first set of instructions would enable the reception and registration of data processing results. The second set of instructions would enable the recovery of data processing results for the frontend display. Finally, the third set of instructions would enable the recovery of data processing specifications and settings.

Data acquisition is often experimental. In some cases, the service captured temporary data points intended for future use. The data that proved to be valuable to the service was usually kept around, but those data points often fell by the wayside if it was not worthwhile. Consequently, these experimental additions and eliminations affected the data model regularly. Both forms of data velocity are problematic for relational databases to handle. Frequently high write loads come with expensive processing costs, and regular data structure changes come with high operational costs. Graph databases address these challenges by having more flexible data models.

The valence of a data point is measured as the ratio of connections to the total number of possible connections. The more connections within the data point, the higher its valence. Over time, highly dense data networks tend to develop, growing both the big data and its complexity. This is significant because dense yet unevenly connected data is difficult to unpack and explore with traditional analytics, such as those based on relational DBMS data stores. In the novel data processing service, the graph database sets the premises for applying artificial intelligence methods. They will be expressed as the most populated path between input data and high-quality results. This will help auto-generate optimal command lines ("paths") for data processing based on quality metrics for the datasets that need to be processed.

5. Conclusions and perspectives

The novel data processing service prototype went through significant changes even in mid-development. Nevertheless, we have successfully prototyped graph data to support automated data processing. While other parts of the data processing service will have to be refactored as part of a long term ongoing project in the laboratory, we could already see that the graph database technology has shown to be efficient, particularly in response to these four data challenges: adaptability of the underlying data model, data volume, data testing, and data valence.

- We have successfully built a prototype microservice based on using a graph database to support fragment screening data processing and analysis. Based on the abovementioned results, the graph data model has been validated as a good fit for the novel data processing service with advantages over the existing relational database system. The graph database technology helped us overcome the rapid change of specific data points and the rapid change of the data model itself. Furthermore, the flexibility of the graph data model allowed us to add new nodes and relationships without compromising the existing network.
- We have achieved a significant step in refactoring the CRIMS infrastructure for fragment screening data analysis through this work. In the future, the use of this technology will allow us to introduce machine learning approaches into our pipelines to optimize automated data processing. For example, the graph database can be queried to unravel the most used and most efficient command lines to run the data processing software.
- As a consequence of this study, future versions of CRIMS will integrate the graph database technology for data processing and will probably be extended to other areas. Additionally, the novel infrastructure sets the premises for integrating other data processing software other than Pipedream.

Consequently, a graph database has shown superior results over a relational database in this application, providing appropriate performance in terms of volume and velocity and excellent results in testing [41]. Nevertheless, understanding how NoSQL databases overcome these challenges is only the prelude to finding the proper database implementation for a given use case.

6. References

1. Zhou, S.-F. & Zhong, W.-Z. **Drug Design and Discovery: Principles and Applications.** *Mol. J. Synth. Chem. Nat. Prod. Chem.* **22**, (2017).
2. Murray, C. W. & Rees, D. C. **The rise of fragment-based drug discovery.** *Nat. Chem.* **1**, 187–192 (2009).
3. Grey, J. L. & Thompson, D. H. **Challenges and opportunities for new protein crystallization strategies in structure-based drug design.** *Expert Opin. Drug Discov.* **5**, 1039–1045 (2010).
4. Patel, H. M. *et al.* **Quantitative structure–activity relationship (QSAR) studies as a strategic approach in drug discovery.** *Med. Chem. Res.* **23**, 4991–5007 (2014).
5. Zander, U. *et al.* **Automated harvesting and processing of protein crystals through laser photoablation.** *Acta Crystallogr. Sect. Struct. Biol.* **72**, 454–466 (2016).
6. Münzker, L. *et al.* **Fragment-Based Discovery of Non-bisphosphonate Binders of Trypanosoma brucei Farnesyl Pyrophosphate Synthase.** *ChemBioChem* **21**, 3096–3111 (2020).
7. Chilingaryan, Z., Yin, Z. & Oakley, A. J. **Fragment-Based Screening by Protein Crystallography: Successes and Pitfalls.** *Int. J. Mol. Sci.* **13**, 12857–12879 (2012).
8. Stock, D., Perisic, O. & Löwe, J. **Robotic nanolitre protein crystallisation at the MRC Laboratory of Molecular Biology.** *Prog. Biophys. Mol. Biol.* **88**, 311–327 (2005).

9. Steven Howard and Chris Abell. ***Fragment-Based Drug Discovery***. (The Royal Society of Chemistry, 2015).
10. Bollag, G. *et al.* **Vemurafenib: the first drug approved for BRAF -mutant cancer.** *Nat. Rev. Drug Discov.* **11**, 873–886 (2012).
11. Saalau-Bethell, S. M. *et al.* **Discovery of an allosteric mechanism for the regulation of HCV NS3 protein function.** *Nat. Chem. Biol.* **8**, 920–925 (2012).
12. Vonrhein, C. *et al.* **Data processing and analysis with the autoPROC toolbox.** *Acta Crystallogr. D Biol. Crystallogr.* **67**, 293–302 (2011).
13. Blanc, E. *et al.* **Refinement of severely incomplete structures with maximum likelihood in BUSTER–TNT.** *Acta Crystallogr. D Biol. Crystallogr.* **60**, 2210–2221 (2004).
14. Smart OS, Womack, TO, Sharff A, Flensburg C, Keller P, Paciorek W, Vonrhein C and Bricogne G. ***Rhofit***. (Global Phasing Ltd, Cambridge, United Kingdom, 2014).
15. McCoy, A. J. *et al.* **Phaser crystallographic software.** *J. Appl. Crystallogr.* **40**, 658–674 (2007).
16. Ernst, P., Plückthun, A. & Mittl, P. R. E. **Structural analysis of biological targets by host:guest crystal lattice engineering.** *Sci. Rep.* **9**, 15199 (2019).
17. A. Sharff, P. Keller, C. Vonrhein, O. Smart, T. Womack, C. Flensburg, W. Paciorek and G. Bricogne. ***Pipedream documentation***. (Global Phasing Ltd, Cambridge, United Kingdom, 2011).
18. Stephens, Z. D. *et al.* **Big Data: Astronomical or Genomical?** *PLOS Biol.* **13**, e1002195 (2015).

19. Angles, R., Prat-Pérez, A., Dominguez-Sal, D. & Larriba-Pey, J.-L. **Benchmarking database systems for social network applications.** in *First International Workshop on Graph Data Management Experiences and Systems* 1–7 (Association for Computing Machinery, 2013). doi:10.1145/2484425.2484440.
20. Srinivasa, S. **Data, Storage and Index Models for Graph Databases.** in *Graph Data Management* (2011). doi:10.4018/978-1-61350-053-8.ch003.
21. Robinson, I., Webber, J. & Eifré, E. **Graph Databases.** (2013).
22. Dupeux, F., Röwer, M., Seroul, G., Blot, D. & Márquez, J. A. **A thermal stability assay can help to estimate the crystallization likelihood of biological samples.** *Acta Crystallogr. D Biol. Crystallogr.* **67**, 915–919 (2011).
23. Márquez, J. A. & Cipriani, F. **CrystalDirect™: a novel approach for automated crystal harvesting based on photoablation of thin films.** *Methods Mol. Biol. Clifton NJ* **1091**, 197–203 (2014).
24. Svensson, O., Malbet-Monaco, S., Popov, A., Nurizzo, D. & Bowler, M. W. **Fully automatic characterization and data collection from crystals of biological macromolecules.** *Acta Crystallogr. D Biol. Crystallogr.* **71**, 1757–1767 (2015).
25. José A. Márquez. **Online Crystallography: Automated pipelines for protein crystallography and drug design.** (2021).
26. Russ Unger and Carolyn Chandler. **A Project Guide to UX Design: For user experience designers in the field or in the making.** (Voices that matter, 2012).
27. Cohn, M. **User Stories Applied: For Agile Software Development.** (Addison Wesley, 2004).

28. Richards, M. **Software Architecture Patterns**. (O'Reilly Media, Inc, 2015).
29. Manastireanu, D. **Software Architecture**. *fxbits.io* <https://fxbits.io> (2019).
30. Curtin, B. **Django Cookbook Web Development with Django - Step by Step Guide**. (CreateSpace independent publishing platform, 2016).
31. Ankur Goel. **Neo4j Cookbook: Harness the power of Neo4j to perform complex data analysis over the course of 75 easy-to-follow recipes**. (Packt Publishing Ltd, 2015).
32. Alistair Jones and Irfan Karaca. **arrows.app: web-based tool for drawing pictures of graphs**. (neo4j-labs, 2021).
33. Regina O. Obe and Leo S. Hsu. **PostgreSQL: Up and Running: A Practical Guide to the Advanced Open Source Database**. (O'Reilly Media, 2017).
34. Reese, W. **Nginx: the high-performance web server and reverse proxy**. *Linux J*. 2008, 2:2 (2008).
35. Matt Stauffer. **Laravel: Up & Running: A Framework for Building Modern PHP Apps**. (O'Reilly Media, 2019).
36. Edwards, R. **Neomodel Documentation**. (2019).
37. **Django REST Framework (DRF)**. (Encode, 2019).
38. Merkel, D. **Docker: Lightweight Linux Containers for Consistent Development and Deployment**. *Linux J*. (2014).

39. Bryce Merkl Sasaki. **Graph Databases for Beginners: The Basics of Data Modeling.** *Neo4j Graph Database Platform*
<https://neo4j.com/blog/data-modeling-basics/> (2018).
40. Thangavelu, M. **The Architecture of Uber's API gateway.** *Uber Engineering Blog*
<https://eng.uber.com/architecture-api-gateway/> (2021).
41. Bryce Merkl Sasaki. **Graph Databases for Beginners: Why We Need NoSQL Databases.** *Neo4j Graph Database Platform*
<https://neo4j.com/blog/why-nosql-databases/> (2018).
42. A. Sharff, P. Keller, C. Vonnheim, O. Smart, T. Womack, C. Flensburg, W. Paciorek and G. Bricogne. **Pipedream wiki page.** (Global Phasing Ltd, Cambridge, United Kingdom, 2011).

7. Supplementary data

7.1. Project source code

The novel data processing source code is available in the following public GitLab repository: <https://gitlab.com/Yorgomoubayed/master-thesis.git>

7.2. Data processing

7.2.1. Data reduction and indexing

The autoPROC framework encloses several third-party programs that help users navigate various steps from images to a fully processed, scaled, and merged dataset. The modules that put together this framework are intended as an offline tool for fully automatic processing diffraction images from experiments. The typical steps during this process involve:

- Image analysis, spot search, and indexing.
- Analysis of diffraction quality and detector parameters.
- Refinement of initial unit-cell parameters.
- Determination of the most likely space group.
- Integration of all images.
- Scaling and merging of integrated intensities.

7.2.2. Molecular replacement

Phaser automates phasing macromolecular crystal structures by both molecular replacement and experimental phasing methods. The phasing algorithms implemented in Phaser have been developed using maximum likelihood and multivariate statistics. The algorithms have proved to be significantly better for molecular replacement than traditional methods in discriminating correct solutions from noise. In addition, the new algorithms give optimal phases with low mean phase error for the phases given by the refined structure for single-wavelength anomalous dispersion experimental phasing.

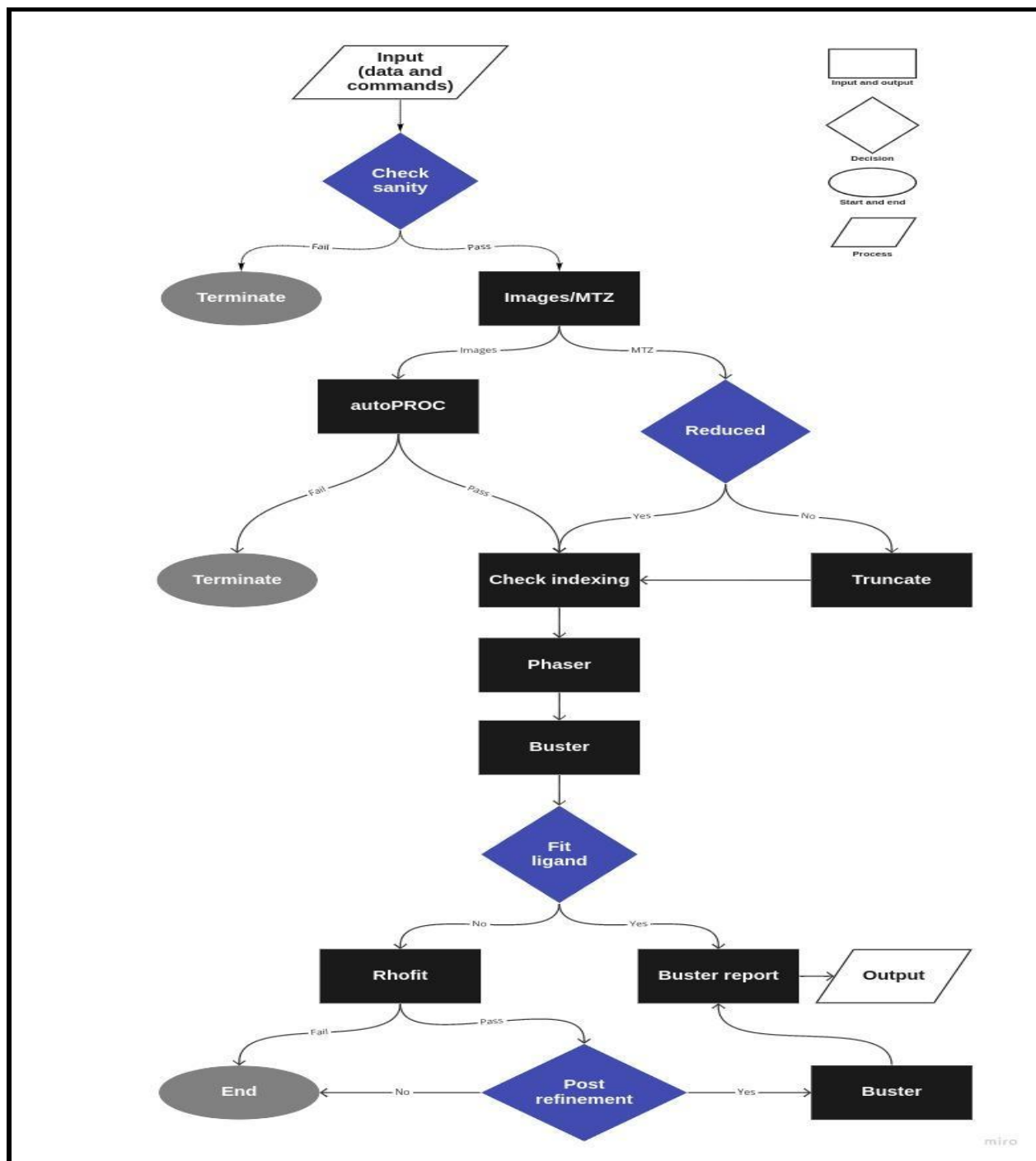


Figure 20 - Schematic of Pipedream's decision tree [42]. The user either inputs unprocessed image datasets or MTZ output from preprocessed image datasets. Image datasets are processed with autoPROC, and MTZ files are checked for reduction. After indexing check, processing goes through Phaser then Buster. If no ligands fit, Rhofit is run until an adequate ligand is found. Once a ligand is fit, the pipeline output is generated. A subsequent refinement step of the optimal ligand solution is performed for better results.

7.1.3. Structure refinement

BUSTER is a maximum-likelihood macromolecular refinement package. It assembles the structural model, scales observed and calculated structure-factor amplitudes, and computes the model likelihood. Additionally, it models the parts of the structure for which an atomic model is not yet available as low-resolution probability distributions for the random positions of the missing atoms. BUSTER handles a variety of cases typical for macromolecular refinement:

- Protein structures with or without ligands or co-factors.
- DNA and RNA structures.
- High- and low-resolution structures.
- Presence of non-crystallographic symmetry.
- Already well-refined structures or structures near the beginning of the refinement process.

7.1.4. Automated ligand fitting

Rhofit is for fitting ligands into different densities. It can change bond lengths and angles within the ligand. The tool enables:

- Searches for correct ring conformations, including macrocycles.
- Searches for the correct chirality if this is not known for an input ligand.
- It uses the gelly geometry function to assess ligand strain and protein-ligand contacts, so its results will be compatible with further BUSTER refinement.
- Runs a large number of independent tests and selects the best solutions that it finds.

Abstract

Recent developments in X-ray-based fragment screening are making significant contributions to the structure-based drug design process. However, they also produce a steep increase in data generated and new needs for data processing and analysis infrastructure. This project aimed to design, implement and validate a new module for automated fragment screening data processing based on a graph database for future integration into the CRIMS infrastructure. This has led to the successful implementation of a novel graph database and associated services, establishing the general concepts and techniques that will enable the expansion and refactoring of CRIMS data processing resources to meet the new requirements. In addition, the service is expected to offer an initial framework into which various other data sources, including additional data processing software, could be incorporated and open the way to implement advanced computing technologies, including machine learning approaches.

Keywords

Crystallography, high-throughput crystallization, structural biology, fragment screening, drug design, drug discovery, data processing, service-oriented architecture, graph databases, RESTful API.